NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AGENT-BASED SOLDIER BEHAVIOR IN DYNAMIC 3D VIRTUAL ENVIRONMENTS

by

David N. Back

March 2002

Thesis Advisor:	Michael Capps
Second Reader:	John Hiles

This thesis done in cooperation with the MOVES Institute

Report Documentation Page

Report Date 29 Mar 2002	**Report Type** N/A	**Dates Covered (from... to)** -

Title and Subtitle Agent-Based Soldier Behavior in Dynamic 3D Virtual Environments	**Contract Number**
	Grant Number
	Program Element Number
Author(s) Back, David	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) Naval Postgraduate School Monterey, California	**Performing Organization Report Number**
Sponsoring/Monitoring Agency Name(s) and Address(es)	**Sponsor/Monitor's Acronym(s)**
	Sponsor/Monitor's Report Number(s)
Distribution/Availability Statement Approved for public release, distribution unlimited	
Supplementary Notes	
Abstract	
Subject Terms	
Report Classification unclassified	**Classification of this page** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** UU
Number of Pages 95	

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Title (Mix case letters) Agent-based Soldier Behavior in Dynamic 3D Virtual Environments	**5. FUNDING NUMBERS**
6. AUTHOR(S) **Back, David N.**	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

ABSTRACT *(maximum 200 words)*

Human behavior in virtual environments is commonly implemented as a finite state machine. This programming approach can be effective and challenging against human players, but its ability to realistically simulate the behavior of cooperative groups of soldiers is limited. This thesis covers the development of an agent-based system to control the behavior of infantry in 3D virtual environments. The system design divides the cognitive process into four modules: perception, mental model, goal decision, and action resolution. Each module attempts to simulate both strengths and weaknesses of human perception and cognition, including instinctive reactions, perceptual error, memory degradation, context-dependent decision-making, and inference. Additionally, the soldiers are influenced by the actions and decisions of the agents around them, enabling cooperation. The resultant agent system was incorporated into a game-like interface and compared to a similar commercial game with standard AI. Overall, 72% of the test subjects thought that the agent behaviors were *Mostly Realistic* or *Totally Realistic*, and 81% found them to be equal to or better than those in the commercial game.

13. SUBJECT TERMS Army Game Project, Artificial Intelligence, Human Behavior Modeling, 3D Virtual Environments	**15. NUMBER OF PAGES** 95
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

AGENT BASED SOLDIER BEHAVIOR IN DYNAMIC 3D VIRTUAL ENVIRONMENTS

David N. Back
Lieutenant, United States Navy
B.A., Stanford University, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION (MOVES)

from the

NAVAL POSTGRADUATE SCHOOL
March 2002

Author:	David N. Back
Approved by:	Michael Capps, Advisor
	John Hiles, Second Reader
	Rudy Darken, Academic Associate Modeling, Virtual Environments, and Simulation Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Human behavior in virtual environments is commonly implemented as a finite state machine. This programming approach can be effective and challenging against human players, but its ability to realistically simulate the behavior of cooperative groups of soldiers is limited.

This thesis covers the development of an agent-based system to control the behavior of infantry in 3D virtual environments. The system design divides the cognitive process into four modules: perception, mental model, goal decision, and action resolution. Each module attempts to simulate both strengths and weaknesses of human perception and cognition, including instinctive reactions, perceptual error, memory degradation, context-dependent decision-making, and inference. Additionally, the soldiers are influenced by the actions and decisions of the agents around them, enabling cooperation.

The resultant agent system was incorporated into a game-like interface and compared to a similar commercial game with standard AI. Overall, 72% of the test subjects thought that the agent behaviors were *Mostly Realistic* or *Totally Realistic*, and 81% found them to be equal to or better than those in the commercial game.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS

2D	Two-dimensional
3D	Three-dimensional
AI	Artificial Intelligence
AGP	Army Game Project
AJROTC	Army Junior Reserve Officer Training Corps
CCTT	Close Combat Tactical Trainer
FPS	First-Person Shooter
FSM	Finite State Machine
ISAAC	Irreducible, Semi-Autonomous Adaptive Combat
ML	Machine Learning
ModSAF	Modular Semi-Automated Forces
OODA	Observe, Orient, Decide, Act
OpFor	Opposing Force
PC	Personal Computer
SOAR	State, Operator, and Result
TACAIR	Tactical Aircraft

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This thesis is dedicated to the Office of the Assistant Secretary of the Army for Manpower and Reserve Affairs, who provided funding. Additional thanks go to:

- Dr. Michael Capps for supporting my rapid and unpredictable schedule during the last months of the thesis.
- John Hiles for giving me my first taste of agent-based systems and inspiring my interest in pursuing them for a thesis.
- Jesse McCree for putting in overtime to design a Return to Castle Wolfenstein™ level so I could get my trial done on time.
- Evan Champlin for helping me during the trial in every way possible, allowing me to do the work of many days in the time I had allotted (one day).
- Maj. Pugh, AJROTC instructor at Seaside High School, without whom I would not have had a ready-made subject pool for my experiment.

Lastly, a very special thank you goes to my wife Angie, our daughter Isabella, and our new baby Samantha who was born during the final throes of this thesis. I will have more time with all of you now that this is done!

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. THESIS STATEMENT

Agent-based decision processes using imperfect perceptions and knowledge can produce realistic soldier behaviors in a detailed, dynamic, three-dimensional virtual environment.

B. MOTIVATION

In simple virtual worlds, the artificial intelligence (AI) can rival and even outperform human players because it is capable of rapid searches of decision trees to determine the appropriate reactions. In an 8 x 8 discrete chess environment with relatively simple rules and perfect information, this approach can be very successful. Even relatively inexpensive commercial chess programs can reliably defeat all but the most skilled human players.

However, modeling human behavior in detailed, dynamic virtual environments is much more complex, especially when dealing with groups of interacting individuals rather than just one-on-one. There are a number of methods used to solve this problem, the choice of which is largely dependent the purpose of the simulation.

Although there are a number of ways to reduce this complexity to a manageable level, they are usually combinations of two elements: first, techniques that reduce the number of options available at each decision point; and second, those which reduce the number of decision points under consideration. Some methods for implementing the first option include the use of heuristics to prune the decision tree and reducing the size of the domain to decrease the number of available options. The second can be overcome by limiting recursion depth for planning algorithms or increasing the complexity of scripted actions so fewer decisions need to be made. Additionally, reducing the number of independent characters that operate in the virtual environment can loosen both constraints.

First-person-shooter (FPS) games use all of these methods to increase the speed and precision of their AI calculations. They operate within fairly narrow domains, limiting the number of possible interactions and perceptions to increase the depth with which they can process them. They are commonly reactive, doing no searching or prediction before choosing an action or series of actions. Additionally, they often use ‘spawning’, to simulate large numbers of independent characters while only having a small number in the game at any time. Furthermore, the AI typically attempts to challenge players by "cheating", giving the AI characters superior speed, accuracy, or perceptions.

Some effort has been made to increase the depth and complexity of FPS game AI programming, adding more academic AI techniques to improve the actual cognitive performance of the game characters. One such system (Laird, 2000) even mimics behaviors learned from expert human players. However, these techniques do not deal with increasing the fidelity of the behaviors; they focus on finding optimal solutions to movement and firing, using the best data and processes available to arrive at their decisions to present the greatest threat to the players.

Providing a challenge for human players is not the only motivation in game development. As virtual environments become richer and closer in representation to the real world, players will expect entities that look like people to act like people as well. In an environment where player-controlled and AI-controlled avatars mix freely (EverQuest™, Unreal Tournament™, etc) and have identical graphical representations, it becomes even more important that the AI act as realistically as possible.

In a training or simulation environment, the importance of this becomes paramount. Training against (or with) an AI is only valuable if the AI acts and reacts in the same way as the people with which the trainee is going to be working with in the real world. The trivial way to solve this is to have players control all avatars in the virtual environment, but this is usually impractical due to constraints on time, geographic location, or politics; for example, real-world combat practice against the forces of a hostile nation is unlikely outside of a state of war.

Thus, there are many situations where human-like behavior is at least as important as intelligent (optimal) behavior. A further discussion of the requirements and complications of humanlike behavior follows.

C. PERCEPTUAL AND COGNITIVE IMPERFECTIONS

Any effort to represent human behavior in a virtual environment requires a model of the process of observing the world and synthesizing information. The model's decisions and actions can be explicitly scripted and unresponsive to the changing environment. Alternately, the AI may be sensitive to all aspects of the virtual environment and make all decisions in response to what it discovers. Most systems are a combination of the two, responding to a subset of the state of the virtual environment based on simulated senses, and reacting with a combination of planned and heuristically scripted responses.

It is relatively simple to imbue virtual entities with unlimited knowledge of the environment. While this can be used effectively to increase the challenge to a human player in the context of a game, it is unlikely to be an accurate representation of 'human' behavior any more than incredible speed, damage resistance, perfect accuracy, or any other trait that is beyond the abilities of models controlled by a human player. Thus, some method of modeling the imperfect acquisition and retention of information about the environment is required.

Having perfect environment knowledge gives an AI a significant advantage over a human player who must use only the information provided by the interface. The effects of imperfect field of view, aliasing effects, incomplete sound spatialization, and variable rendering rates undercut the player's ability to accurately perceive the virtual environment. Most simulations offset this in part by providing additional on-screen cues or sounds to indicate significant events or objects.

AI-controlled models hobbled to an equivalent level would put them at a disadvantage because humans are adept at "filling in the blanks" when presented with imperfect information. If the AI cannot make inferences about the environment that go beyond what it directly perceives, it will be ineffective in simulating human behavior. Furthermore, a human's ability to use inference will effectively increase as the environment's fidelity increase; that is, as it becomes closer to a real-world environment, the more pertinent the player's background knowledge becomes. Thus, some method of mimicking inference is necessary for the AI in the absence of actual cheating.

D. AGGREGATE AND INDIVIDAL CONTROL

Unlike the real world, it is perfectly reasonable for an AI system to have a global entity that directly controls many of the virtual characters in a simulation. Such a scheme reduces the strain on the system's computing resources when compared to entity-level control, allowing for more detailed group behavior and greater depth of planning. This technique is common in military simulations that model the behavior of groups of soldiers instead individually managing their actions and interactions. Such systems give the designer explicit control of desired interactions between entities and groups, allowing complex group behaviors learned and tested in real life to be modeled without having to encode the behaviors and interactions required into each individual.

Alternatively, unscripted interactions between simpler entities can also be used to produce complex behavior, as is shown by systems like ISAAC (Ilachinski, 1997). Individually, such software agents have simple actions and behaviors, yet as a group they display "emergent behaviors" that appear much more intelligent than the simple agent structure would suggest. Although intriguing, it takes significant experimentation to define the agent qualities required for a desired group behavior. More commonly, such systems are used as virtual battlefields to gain insight into the gestalt effects of new individual capabilities.

E. NON-DETERMINISTIC BEHAVIOR

The indeterminacy and complexity of human behavior is a problem for modelers. Psychology has yet to produce a completely valid and tested model of real human behavior, so there is little wonder that simulating it on a machine is problematic to an even greater degree given the significant differences between human and computer thought process.

One way that has been tried (Jones, 1999), often with great success, is to create a list of rules and contingencies that is sufficiently long and complex that it cannot be easily anticipated or predicted by a human observer. In practice this is quite difficult; even for a discretized, finite-state game like chess. A supercomputer is required to beat a human champion. In a situated simulation environment (like TACAIR-SOAR), the rule-set must be much more complex to properly represent human behavior. Furthermore, a complex environment and continuous time preclude any possibility of creating a 'spanning set' decision matrix which contains all possible actions at every possible breakpoint, something that is possible in turn-based games like chess. Thus, the designer's notion of what actions are possible, likely, or beneficial will color the content of the AI's rule-set.

As might be guessed, more complex sets of rules and conditional checks greatly increase the need for storage space and processor time to conduct the searches for the appropriate behavior. This in turn increases the length of the decision cycle and decreases the system's ability to respond to rapidly changing environments. Furthermore, it occupies processor resources that could otherwise be used for other processor-intensive tasks (e.g. screen rendering, physics, collision-detection).

An obvious solution to this would be to introduce randomness into the decision process to simulate the complexity that cannot be properly modeled. Careful design and testing is required to ensure that the random factors cannot produce unrealistic results. The AI must strike a balance between true deductive decision-making and subtle nudges with pseudo-randomness.

F. FIDELITY

Realism, or model fidelity, is a high goal indeed, albeit one that is at present limited by the technology available. Since the complete contents of a real-world environment cannot presently be modeled, the designer has to choose a subset of the possible affectors and environmental cues to include in the simulation.

The problem domain drives the need for fidelity. For a game, realism is only necessary insofar as it promotes enjoyment; a completely abstract game like Tetris™ only requires the minimum level of fidelity to the physical phenomena it appears to be modeling (falling geometric objects). As an environment becomes more like real life, players will reasonably expect higher fidelity in the physics, biometrics, behaviors, and other perceptible factors. For a training simulation, though, fidelity is absolutely required for those environmental factors that would affect the trainee's decision process.

This raises two very complicated problems: first, how to discover what information is required; and second, how to represent it sufficiently that the human players will respond appropriately. In a FPS environment, the main way that the entities pass information is through their choice of actions and how these actions are presented to the other entities and human players. Thus, which actions an AI chooses determine what it does in the environment in addition to how the players see, understand, and react to it.

G. GAME CONSTRAINTS

Generating an agent system that mimics the human decision process would be an admirable achievement in and of itself, but if it is not entertaining and easy to interact with, it will not be used. This is doubly true of a FPS game environment. Since games dominate the world of 3D situated virtual environments, and since many comparable AI systems have been made in conjunction with games, it is natural to do AI experimentation in a game environment (Adobbati, 2001). Experimentation with such systems is also very practical for a number of reasons. They are designed to run on personal computers, which are easily accessible and relatively inexpensive. Most FPS games are designed to be easily modifiable, often including a scripting language simplify changes to high-level behaviors without having to worry about the underlying mechanics of the engine. Finally, using an existing game engine alleviates the need to design physics, graphics, animation, memory management, and other necessary code before starting the intended work on the AI and behaviors.

Nevertheless, use of an existing game engine imposes certain constraints on the design of the AI. Some major points of a production game are fun, graphic interest, and rendering speed. Making a game fun may be in direct conflict with making it realistic, depending on the specific content of the game. High-end graphics and high frame rate are both heavy consumers of processor time, reducing the amount that is left for the AI.

Therefore, using existing game development tools presents a trade-off between their usefulness and the additional constraints they impose.

H. GOALS AND METHODOLOGY

The design goal of this system is to create an agent-based controller for Unreal entities in the Army Game project that demonstrate believable humanlike actions and decisions. This entails using the Unreal engine facilities and following the production goals of the Army Game Project.

The project begins with a review of similar simulations and games to discover any useful techniques which might be adapted directly into the AGP. Additional research into agent systems will give the conceptual framework in which to design the AI for the AGP soldiers. Concurrent study of the Unreal™ engine and the UnrealScript™ programming language will lead to an understanding of the capabilities and limitations inherent in the system. Then, an overall design of the information and decision flow will result in a notional block-diagram of the required code modules and functions that will be required. After the framework of the information-decision-action cycle is encoded, it will be incrementally tested and expanded to increase the repertoire of behaviors and circumstances it will support. Finally, the resultant system will be tested against a game with traditional AI control to demonstrate the strengths and weaknesses of agent-base design, and suggest directions for future development and research.

I. THESIS ORGANIZATION

The remainder of this thesis is organized as follows:

- **Chapter II: Background**. Research in various areas similar to the Army Game Project, either through gaming, agent-related work, AI, or military simulation. Description of existing and past research, which areas were complementary, and which were inapplicable.
- **Chapter III: Design**. Describes the underlying methodology and structure of the agent system, and discusses the decisions and trade-offs made to tailor the design to the implementation and purpose of the AGP.
- **Chapter IV: Implementation.** Describes the basic structure of the Unreal™ engine and the data structures on which it is built. Then details the modules created for this thesis, how they interact with the underlying code, and how they implement the decisions made in Chapter III.
- **Chapter V: Experiment and Analysis**. Experimental design for testing of the system and analysis of how well it met its design goals. Conduct of the actual experiment, and what results were derived.
- **Chapter VI: Conclusions and Future Work**. Discussion of the strong and weak points of the system as currently implemented, and suggested directions for future research and development.

THIS PAGE INTENTIONALLY LEFT BLANK

II. RELATED WORK AND BACKGROUND

This chapter presents an overview of several systems whose presentation or content are similar to the Army Game Project. It discusses their similarities and differences, and which of their methodologies or implementations are applicable to this thesis.

A. INTRODUCTION

There is a broad range of systems attempting to simulate the behavior of the modern soldier on the battlefield and display it. They can be broadly organized into four categories, according to purpose. These are military constructive simulations, military training simulations, game simulations, and agent simulations. In the case of games and agent simulations, only those that work in the military domain will be considered.

For any of these to be useful for the purposes of this thesis, they must be scaled to individual dismount infantry, use a graphical, 3D interface with a human player in real time, and they must simulate the behavior of real soldiers. A system without one or more of these aspects may still be useful from a conceptual standpoint, but it will not be an answer to the basic problem.

Table 1 below summarizes how the categories of simulations compare to the requirements of the Army Game Project. The remainder of this chapter discusses these analyses and their consequences in further detail.

Metric	**AGP Requirements**	**Military Constructive Sims**	**Military Training Sims**	**Games**	**Agent Systems**
Time	**Real-time**	No	**Yes**	**Yes**	No
Graphics	**3D**	No	**Yes**	**Yes**	No
Presentation	**First-person**	No	**Yes**	**Yes**	No
Scale	**Entity-level**	No	**Yes**	**Yes**	**Yes**
AI Behavior	**Realistic soldier**	**Yes**	No	**Partial**	**Partial**
Control	**Individual**	No	**Yes***	**Yes**	No

***** individual vehicles only; dismount infantry are typically handled in groups

Table 1. Comparison of Existing Systems with AGP Requirements

B. MILITARY CONSTRUCTIVE SIMULATIONS

Military Simulations are used extensively to predict possible outcomes for conflicts on the battlefield. There are a number of simulations in current use by the U.S. military, including ModSAF (Calder, 1993), Janus (Ramsey, 2002), and the under-development Combat XXI project (Denny, 2001). All of are based on extensive real-world data and research, so they are as true to reality as possible, within the constraints of the simulation. Since they are used to model and predict the effects of real-world warfare, model fidelity is of the utmost importance.

However, they are also typically of much larger scale than individual soldiers, often not considering anything sma ller than regimental scale; variations of Lanchester equations (Taylor, 1980) are used to average out the behaviors of the large numbers of troops within each atomic unit of military force. As such, the behaviors that they model are not particularly applicable to individual soldier modeling, since the extreme and abnormal behaviors that such equations ignore are of particular interest in a situated first-person game. Furthermore, the interface with the human participant is typically a top-down "god's eye" view, instead of one situated within the simulated environment. Usually these simulations run in an accelerated time-scale, since they are designed to run many iterations of the same scenario as quickly as possible for statistical analysis of the results.

Therefore, although they model human soldier behavior as accurately as possible based on historical data, the differences in scale and presentation make such modeling inapplicable to the AGP.

C. MILITARY TRAINING SIMULATIONS

Although there are numerous training simulations used by the military, there is little reason to consider those used by the Air Force or Navy when considering applicability to the AGP. The Army's primary tactical training simulation is the Close Combat Tactical Trainer (CCTT) (Foster, 2002), a system that links together numerous cockpit mock-ups of tanks and armored vehicles with a realistic, first-person, real-time virtual environment presented graphically through the cockpit windows. The presentation is very similar to that of the AGP, although there is much more hardware involved in correctly modeling all of the requisite systems and controls in a military vehicle; the AGP assumes that the user will only have a keyboard and monitor for input and output.

The scenarios, like the constructive simulations mentioned above, are designed to be a realistic as possible, involving actual elements of the U.S. and foreign arsenals; their physical and combat models are similarly as accurate to real life as possible.

However, CCTT suffers from generally poor or nonexistent AI. A crew of human trainees runs each vehicle. Dismount infantry are controlled as squads rather than individuals, and a console operator handles each squad. All forces must be controlled directly by a human operator since there is no behavioral AI to allow opposing forces to act with autonomy. As such, although very similar to the goals of the AGP, the CCTT has very little to contribute to the formation of the AI module.

D. COMPUTER GAME AI

Existing game AI in the first-person-shooter genre is very promising. Various games like Unreal, Quake, and Doom have had AI to control most of the opponents characters in the game for many years, and they use the correct scale, time frame, and presentation. The method in which game AI is accomplished deserves additional description.

Nearly all modern games control their characters through a finite state machine (Funge, 1999). Each character in the game is in a state that corresponds to some goal or attitude (‘kill’, ‘run away’, etc). When in that state, their actions are programmed through a sequence such as ‘find nearest enemy, point at it, shoot, repeat’. Transitions from one goal to another are conditioned by queries to internal properties (e.g. health) and external environment (e.g. surrounded, enemy has better weapons). Complexity is achieved by refining the decision process for switching between states, improving the behavior within each of the states, and increasing the number of states and thereby the complexity of the logic to jump between them.

Generally, this approach yields reasonable results. As the code grows longer and more detailed, it becomes more of a challenge for the human players who must oppose it.

Even so, in order to present a challenge, the computer players usually better aim, speed, resistance to damage, or vastly superior numbers. When on completely equal terms, AI only presents a challenge to human players in simple environments like chess. Furthermore, the goal of challenging the players is often in opposition to realism, since many advantageous behaviors in the game (e.g. spawn-camping, pickup-camping, using explosives to jump farther) are only possible because of unrealistic aspects of the game environment.

More importantly, the finite state structure is not as attractive as an agent system (detailed below) for intuitive representation of the goal decision process, and it is necessarily less flexible since all state transitions must be explicitly encoded into each state. Nevertheless, the structure of a traditional game AI is important because the underlying base of the Unreal™ engine is designed for this style of control. Whatever solution is used, it must interface with the Unreal™ engine, and the higher level at which this occurs, the less difficulty there will be in programming and integrating.

E. AGENT SYSTEMS

Agent systems are a growing specialty in the overall discipline of artificial intelligence. Conceptually, such a system uses the interactions between numerous independent software entities or "agents" to achieve complexity and emergent behaviors. Behavioral emergence occurs when a group of agents produces an aggregate behavior that was not explicitly encoded and usually was not expected by the designer. It is a product of the numerous interactions between the agents, and has been shown to generate apparently intelligent behavior even with relatively simple agent structure.

Agents are an extension of object-oriented programming. Objects have data and methods that may be invoked by external callers. Agents add a layer of intent. Instead of merely having their functions called, the agent responds to stimuli within the virtual environment to decide which of its methods it wishes to use. Agents usually have limited perceptions, restricting their information about the environment to a subset or perturbation of the truth. The agent's current goal is chosen by judging all possible goals at each juncture according to an heuristic weighting function which takes the agent's perceptions of the environment and its own internal characteristics as arguments. When selected, a goal then chooses and executes an appropriate action. When the results of the action are perceived, an objective or fitness function then allows the agent to complete the feedback loop and determine if its chosen actions have improved or harmed its overall success in the simulation, thus allowing it to gradually modify its overall behavior in favor of those actions that produce the best results.

Agent systems have been applied to numerous problem domains. In addition to significant crossover with robotic control and machine learning (ML), agents have been used for simulations of human organizations, predators and prey, and various sports and games (Stone and Veloso, 2000). For example, a significant body of research has been put towards creating teams of agents to play soccer on a virtual field using various learning techniques to improve their abilities over time (Stone, 2000).

However, the seminal military-themed agent system was Andrew Ilachinski's ISAAC system. A proof-of-concept rather than a true simulation of battlefield behavior, ISAAC pitted an army of blue squares against an army of red squares in a discretely-gridded world, with each army attempting to capture the enemy's flag (a specifically marked grid for each side). Each agent was described by a vector of weights indicating its propensity to move toward or away from each feature of the environment: friendly agents, injured friendly agents, healthy and injured hostile agents, and each team's flags. During each time interval, each agent observed its perceptual world (a user-defined number of squares around their own location). Based on its perceived environment and its own personality, it decided which of the eight surrounding grid squares it would attempt to occupy. Once all agents had decided on their actions, external rules adjudicated conflicts and moved the agents appropriately.

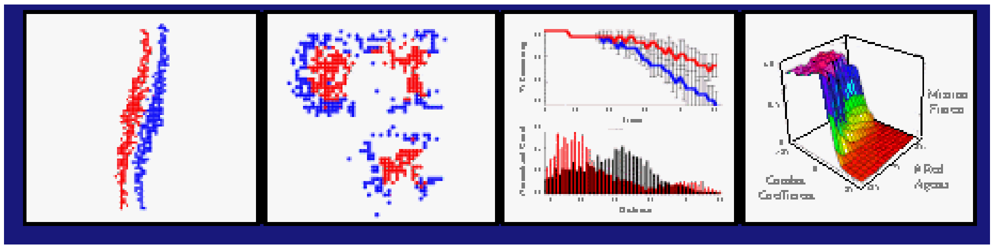

Figure 1. ISAAC Display and Statistical Output (Ilachinski, 1999)

Although the design of the individual agents and the virtual environment was very simple, when grouped together the agents performed unexpectedly complex behaviors; frontal assaults, flanking maneuvers, delaying actions, and guerilla attacks were all noted with various modifications to the team size, weight vector, sensing range, and weapon range and effectiveness. None of these behaviors were explicitly coded into the agents, nor did any single agent control the actions of any other.

More complex systems have followed ISAAC that increase or modify its behaviors to simulate more complicated environments or behaviors (Ilachinski, 1999). However, none of them to date has changed the fundamental 2D nature or essential simplicity of the agent's environment. Such systems do not tend to run in real time, either because they are too abstract for real time to be a meaningful concept or because they are designed to do as many runs in as little time as possible to gather statistical results.

The strength of agent systems comes with the interactivity between large numbers of fairly simple individuals. It would therefore be inappropriate (and fairly boring) for a human player to directly control a single agent within the simulation. They would have very few parameters to control, and they would not be able to influence the outcome of the simulation directly, since they would not be able to exert control outside their single agent.

On the whole, the goal decision process that agents use is an intriguing alternative to the finite state structure described above. Because it directly involves the goals and intentions of the agents as well as allowing interactions between groups of agents in a cooperative action, it conceptually points to a similar approach in the more detailed environment of the AGP.

F. CONCLUSION

None of the existing artificial intelligence technologies discussed will support the combination of features needed for the AGP. However, the applicable elements of finite state game AI and the more intuitive goal-action decision structure of agent systems are both useful. The AGP design will incorporate many features of each, combining the presentation method, 3D environment, and state-based actions of the Unreal™ engine with the goal weighting and dynamic decision-making from agent systems.

Unfortunately, the military simulations had very little to offer, as their strengths lie more with the accurate modeling of physical quantities and large group behaviors rather than individual motivations and actions. Thus, despite having the same genre and context as the AGP, they offer no significant source of inspiration for this project.

THIS PAGE INTENTIONALLY LEFT BLANK

III. ARCHITECTURE

A. INTRODUCTION

Chapters I and II alluded to many desirable characteristics for a system designed to simulate individual and group soldier behavior for the Army Game Project. This chapter presents the motivations, theory, and design of the AI module that will accomplish this goal. The concepts and data structures included are, to the degree possible, implementation-independent. The specifics of how they were adapted to the Unreal™ engine will be discussed in Chapter IV.

B. REQUIREMENTS

Primarily, the AI module should produce humanlike behavior that is believable to a naïve observer (such as a non-military person playing the game for the first time). A more advanced goal would be to have it be convincing enough for subject-matter-experts to accept the agents' behaviors as natural. This is, of course, highly subjective, and thus cannot be rated on any fixed scale. More detailed analysis of the 'humanity' of the AI module is included in Chapter V.

In addition to this main goal, and in support of it, the module will have to run quickly enough that it does not interfere with game play (rendering, user input and feedback, etc) or 'think' too slowly to properly respond to changing conditions in the virtual environment. Adjunct to this is that it be as memory-efficient as possible.

Overall, the design reflects a series of improvements to the information receipt and processing of each soldier. As a whole, these alterations will improve the realism with which the soldier interacts with its environment, especially with other soldiers (AI or player-controlled).

C. CONCEPTUAL DESIGN

The basic structure of a system to control a virtual entity (a soldier in this case) will have to generally follow the 'perceive-decide-act' paradigm (or as used by the Army, 'observe-orient-decide-act' or OODA). This model of information and decision flow leads to a four-module design to interact with the virtual environment, as shown in the figure below.

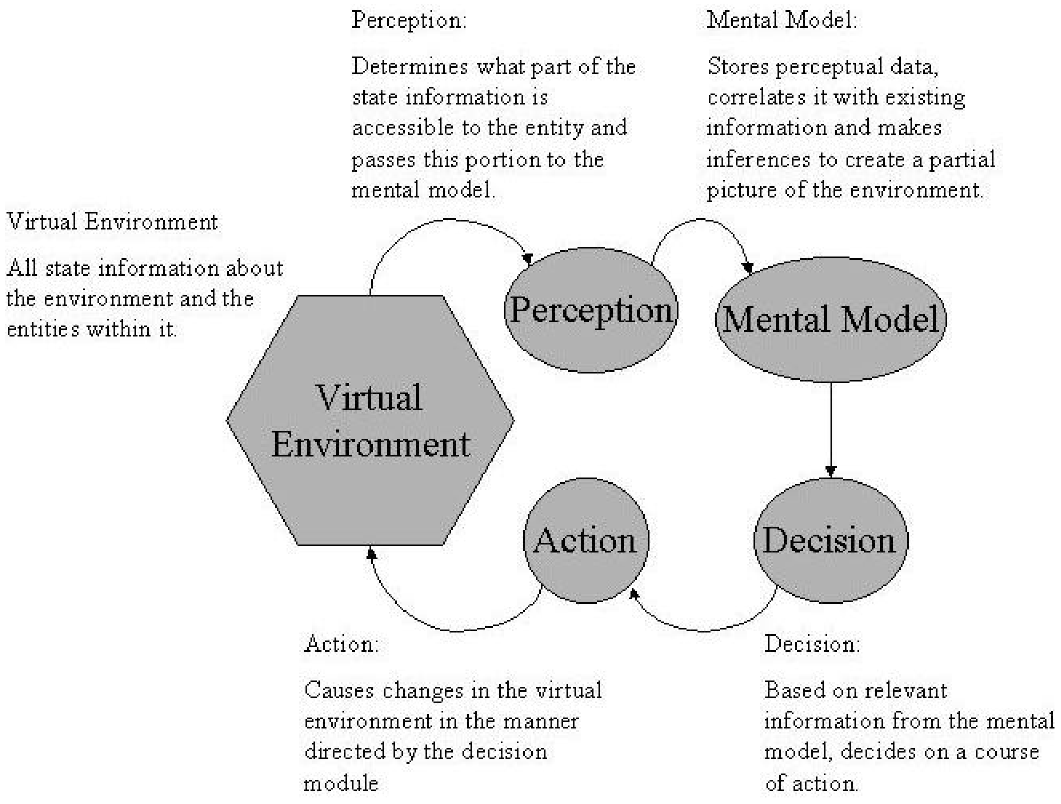
Perception:
Determines what part of the state information is accessible to the entity and passes this portion to the mental model.
Mental Model:
Stores perceptual data, correlates it with existing information and makes inferences to create a partial picture of the environment.
Virtual Environment
All state information about the environment and the entities within it.
Perception
Mental Model
Virtual Environment
Action
Decision
Action:
Causes changes in the virtual environment in the manner directed by the decision module
Decision:
Based on relevant information from the mental model, decides on a course of action.

Figure 2. Four-Module AI Design

The following sections discuss each of these modules in further detail, especially focusing on potential implementation decisions, problems, and possible solutions. Interactions between the modules and between entities are also discussed.

1. Perception

There are several specific design goals for a perception module to make it mimic (as much as is possible) the way in which a human player perceives the virtual environment. This is necessarily difficult since in many ways it involves decisions about what to limit; for example, a player can rarely perceive more than 45 degrees of horizontal arc because of the limitations of a screen, whereas a virtual entity can perceive as much of a visible arc as is desired. Should the AI be limited to the same perceptions as a human, or should they have the same field of view as a human in the real world (around 180 degrees)?

Despite these ambiguities, there are certain design goals that are fairly unarguable. The first of these is that the amount of information gained about a particular object sighted ought to be dependent on the range and intervening cover. What this dependence should be is subject to testing.

Furthermore, knowledge of an object should be particulated in such a way that just ‘seeing’ or ‘hearing’ something does not immediately impart complete knowledge of its nature. Partial knowledge is the natural way to allow for misconceptions and errors in decision. This constraint will affect how the mental model is designed as well.

Additionally, a humanlike perception system will have to allow for error. Even if partial information is missing, being able to rely on the details already gleaned about a target greatly improve the AI’s chances as compared to a human player (who may mistake enemy camouflage for friendly, for example). Some system of “spoofing” the perception system, both inadvertent and deliberate, is necessary.

Finally, there should be some dispensation for reflexive behavior. Actions like ducking in response to a surprise or loud noise are hardwired into the nervous system and are not subject to decision or deliberation. Modeling these sorts of reactions should provide an additional element of realism, since the players will have immediate feedback that the AI is responding to their actions in an appropriate way. Since such reflexes occur independently of the cognitive processes, it makes sense to implement them as a direct connection between the perceptual module and the appropriate actions, bypassing the goal decision process.

2. Mental Model

Human players base their decision on more than just the immediate perceptual landscape. They have the advantage of a mental representation of the level, allowing them to plan ambushes, escapes, resupplies, and other courses of action that are impossible without knowing information outside of perceptual range. A complete mental model of all entities seen or heard in the world would be the best answer, but likely prohibitive in terms of memory and processor time required to support queries.

In an infantry-oriented game like the AGP, the soldiers exhibit the greatest degree of dynamism and have the greatest impact on the course of the game. Therefore, the highest priority is to keep track of where and when they were perceived, who they were, and what they were doing.

Information is perishable, for two reasons. First, information like location, velocity, and current goals are dynamic. Therefore, the mental model should degrade its knowledge of properties not being directly observed to the status of a ‘guess’, making them less reliable. In addition, memory is not perfect. Even static qualities like rank, entity type, or alignment may not be perfectly recalled after some time has passed, especially if there are numerous contacts stored in the mental model. It should be possible to confuse memories and interpose one perceived being with another.

Inference is a difficult problem, but solving it (or at least simulating it) is necessary for human behavior simulation. As an example, the mental model must be able to see a single soldier and posit the existence of the rest of the squad, knowing that soldiers tend to work and travel in groups. Inference should be more fallible than direct observation, for obvious reasons, and should be dependent on prior knowledge of the enemy’s order of battle and army structure. A more advanced goal would be to have the AI learn about the structure of the enemy’s army (and correct mistakes in the preconceived notions) based on observation, but this is probably beyond the scope of the AGP considering that most AI-controlled soldiers will not persist for longer than a single mission, certainly not long enough to make a complex learning system worthwhile.

The concept of confidence is a good way to address both the problem of inference and perishability of information, as well as encoding various personality traits. While distinct from the actual accuracy of a given piece of information, the mental model’s confidence in that data can be used as a weighting factor on its influence. This produces two kinds of errors, both of which are desirable. First, if insufficiently confident of a piece of data that is nevertheless true, the AI may hesitate or fail to act appropriately while waiting for new information, especially if operating under restrictive prior assumptions (only to fire in self-defense, for example). Conversely, excessive confidence in perceived or guessed data can lead to inappropriate actions (friendly fire, running away from a single soldier thinking he is an entire squad, and so forth). While the capacity to do these things will certainly not optimize the AI's abilities, it will make them more humanlike and open up a wider and more realistic range of tactics for players to use.

Finally, sharing of information between soldiers is a desirable goal. Without it, coordinating actions between multiple soldiers requires absolute direction from a higher-level entity. This may be appropriate under certain circumstances, but it is also in direct contrast to the individualistic and agent-oriented design discussed so far. Unfortunately, propagating information between soldiers will require additional processing overhead to determine which soldiers are close enough to communicate and how much information can be passed in a given amount of time. Thus, some simulation of information sharing is likely to be more useable. Fortunately, the organizational structure of an infantry unit lends itself well to removing some of the difficulties; the soldiers in a squad will generally stay in close proximity, and are trained to communicate with each other. Thus, is may be reasonable to remove communication entirely and assume that all soldiers in a fire team or squad have access to the same information. Their reactions to this information will still vary because they must relate it to their individual location, condition, and tactical situation before deciding on an appropriate goal.

3. Goal Decision

Once there is a picture of the environment, the AI can make decisions about what it wants to do. Soldiers will always have a number of conflicting interests. In addition to the fundamental desire to avoid death, they should also be motivated to attack the enemy, follow the orders given by their superiors, stay with their squad, and accomplish their overall mission. Special circumstances will extend this list; patrolling an area, looking for enemies, defending friends or certain locations, or just waiting for something to happen are all appropriate at certain times.

The combination of the soldier's state and their knowledge of the environment will lead to such a decision. In the spirit of agent systems, this decision should be designed as a weighted comparison between all of the possibilities available at each juncture rather than a pre-scripted jump from one state to another. This should increase the flexibility of the system and allow intuitive encoding of the AI's mental and emotional state rather than having to distribute it throughout the structure of the state-machine. This should permit easier configuration of the attributes of individual soldiers (bravery, aggressiveness, overconfidence, laziness, caution) to create a wide diversity of individual behaviors from a fairly small set of properties and capabilities.

Most goals fall into two fundamental categories. First are the orders, things that the AI decides to do because its ‘leader’ has told it to do so. These can exist at many levels, including direct commands from a squad leader, standing orders, and interpretations of commander’s intent. These can be given various weights that allow one to pick up when another becomes impossible or unclear.

However, in order to simulate human behavior there, the soldier must be able to act instinctively in times of stress. Therefore, there should be another level of behaviors that are responses to external stimuli; threats, perceived vulnerabilities, curiosity, laziness, and so forth. When instinctive motivations are greater than their motivation to obey their current command, they should break discipline and act individually.

4. Actions

Goals are fairly broad representations of the AI’s attitude toward the world and their immediate environment. Once a goal is made active, the AI needs to figure out which specific actions are appropriate to accomplish this goal.

A discrete and countable list of all possible actions is possible due to the necessary limitations of the interface with the chosen virtual environment. However, an intelligently chosen subset of combinations of these elemental actions should suffice to give the agents flexibility enough to deal with the world. There are two main trade-offs involved here.

The first concern is the number of possible actions. A long list would give greater flexibility and the designer would be able to encode more specific combinations of actions that are appropriate to the simulation. However, a larger set of actions requires a more detailed algorithm for deciding which one to use, which in turn requires a more detailed virtual environment on which to base the determination. Furthermore, as the complexity of the set increases, it becomes increasingly likely that certain actions will dominate others due to slight dissimilarities in their weighting schemes, in which case the marginalized actions might as well be absent. This problem can be partly abstracted away through judicious use of random numbers, selecting between otherwise equally applicable actions.

A shorter list of actions for each goal makes programming and debugging far easier, and allows the virtual environment to be simpler. Therefore, reactions are more robust and quicker. This has its own problems though; a short list of possible actions may become predictable, especially since humans are adept at noticing patterns. Ultimately, this could violate the design goals of the project, since humanlike behavior should not seem deterministic.

The second decision is how complex each action should be. Short, concise actions (e.g. 'shoot designated enemy', 'move forward', 'crouch') are easily and quickly accomplished, and it is difficult to interrupt them. They are essentially one step above directly manipulating the engine and animations. However, because of their simplicity they do not individually have much chance of accomplishing a goal. Additionally, short actions require frequent reassessments to determine what to do next. If the AI runs quickly, this may not present a problem and will give the system an opportunity to respond more quickly to changing circumstances. If the AI gets bogged down then the added overhead of frequent reassessment may lead to an unforgivable performance hit and produce delays or stutters in the soldiers' actions.

More complex actions, on the other hand, give the designer the ability to script more complex tactics which would be unlikely to arise from the reweighting scheme, especially those that involve cooperation with other soldiers. As an example, a trained Special Forces team might know an "Australian Peel" maneuver where one soldier covers the rest until he is the last in line, then he retreats while the next soldier covers him. It is unlikely that such a specific tactic could result from primitive actions and reweighting, which is appropriate because real soldiers would have to learn it through repeated practice rather than discovering it on the battlefield.

D. CONCLUSION

The design of the AI system is necessarily incremental, and will continue long after this thesis is complete. The first priority is to create a functional AI that produces simple, believable behavior and provides a proof-of-concept for the information /

decision loop described above. Once this is accomplished, additional complexity will be added. Because the design is modular, new perception modes, mental model parameters, goals, and actions can be added on after the initial design with limited interference with the existing structures.

Table 2 gives a summary of the initial implementation goals for this thesis. Chapter VI will discuss avenues of future work to improve and refine the system.

Module	Initial Design Goals
Perception	• Simulation of sight and hearing • Human field of view (approximately 180 degrees horizontal) • Quantity of information gained dependent on time and range • False information gain possible • Interactivity with mental model • Simulation of reflexive actions
Mental Model	• Representations of other soldiers in the environment • Knowledge divided into individual traits • Knowledge is lost over time • Simulation of inference for simple traits (e.g. location) • Representation of confidence in known traits • Shared mental model by fire team or squad
Decision	• List of "instinctive" goals • Algorithm for assigning weights to each goal • Actions associated with each goal • Algorithm for choosing appropriate action to accomplish goal • Scheme for timely reweighting when the situation changes
Action	• Short list of actions • Medium complexity of actions • Most actions applicable to more than one goal • Feedback to calling goal when action is ineffective

Table 2. Initial Design Goals

THIS PAGE INTENTIONALLY LEFT BLANK

IV. IMPLEMENTATION

A. INTRODUCTION

Adapting the design goals described in Chapter III required extensive modifications to the existing Unreal AI system, introducing a number of new modules in place of the single Bot module used previously. The overall goal was to create an agent system to control individual soldiers and to share information between them. At the same time, due to the constraints on memory, processor time, and programming timeframe, acceptable concessions were made provided they did not have a significant effect on the operation of the system.

The remainder of this chapter discusses the Unreal system as it is currently implemented, gives an overview of the additional AI code written for the AGP, and makes some conclusions about the process. Direct quoting of computer code and references to programming language keywords are signified with a font change, as in class `MentalModel`.

B. THE UNREAL ä SYSTEM

1. General Description

The Unreal™ engine exists on two levels. First, there is the underlying code base, written in C++. These modules provide for the fast manipulation of processor-intensive elements of the virtual environment: rendering, raytracing, collision, textures, animation, networking, and other areas that must be done quickly to be done well.

The upper level of the code is in a format called UnrealScript™. This is a programming language created entirely to interface with the Unreal™ code base. It has many commands and control structures specific to the engine in addition to constructs and operators normally available to a programming language. It communicates with the C++ code base in two ways. First, script functions can be written to be triggered by predefined events generated by the code base. Conversely, script can be made native, which leaves the variables and function headers in the script, but requires the implementation to be written in C++. This allows script entities to directly call functions in the quicker compiled code.

The normal mode of script execution is a read-decode-execute interpretation done in real time. This is considerably slower than compiled code, as might be expected. Each line of script takes roughly 100ms to execute, regardless of its contents. This makes brute-force searches and time-stepped code more problematic.

UnrealScript™ is a very strongly typed language, with limited and sometimes prohibited type recasting. For example, a variable of type enum cannot be recast as a byte (its underlying type) to be used in a generic function that might take several different types of enumerated values. This makes code abstraction and extension more difficult; multiple functions that do essentially the same thing must be written instead of a single multipurpose function, and adding types must also include adding functions to handle them.

Despite these problems, UnrealScript is a powerful language for dealing with the Unreal Engine, and has built-in abstractions for many functions that would be intensive and difficult to program at the C++ level.

The next section will discuss the base classes and constructs in Unreal™ and how they behave and interact.

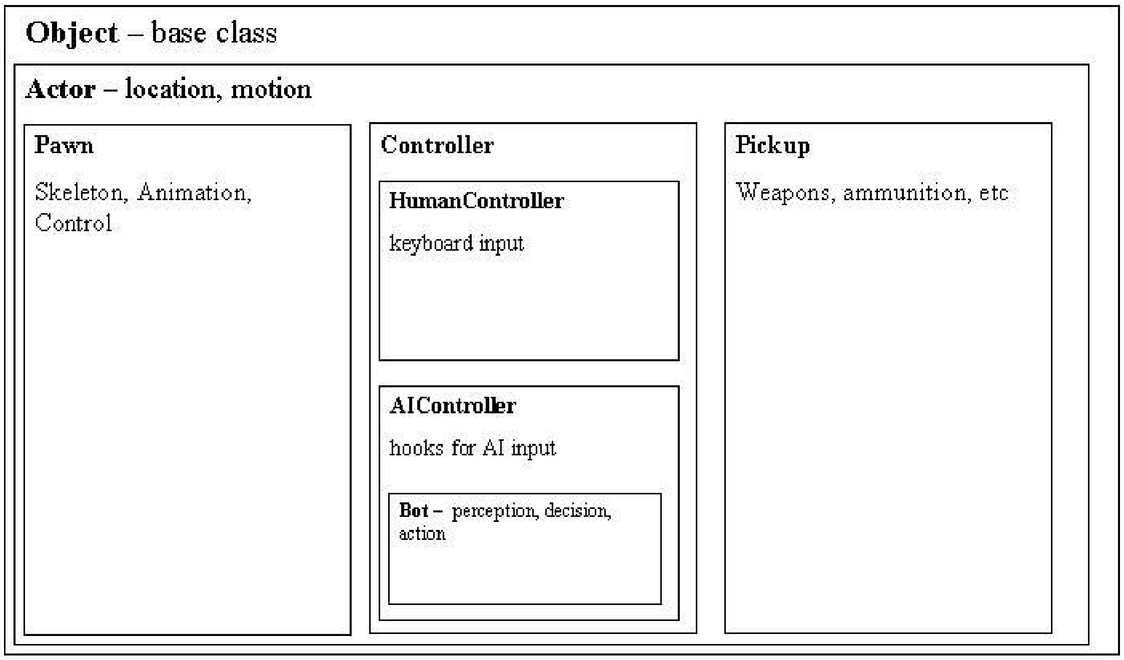

Figure 3. Unreal Class Hierarchy

2. Script Classes

Everything in an Unreal world is built from script classes. The most primitive of these is the `Object`. `Object`s have very limited functionality, and since they do not have any location or appearance they cannot be instantiated in the same way as higher-level classes. Furthermore, they do not interface with the game clock, so it is difficult to manipulate them in timed or periodic ways. Because of these limitations, none of this thesis work was done with `Object`s, despite the relatively low overhead involved.

`Actor` is a subclass of `Object`, and adds significant functionality. It has location and velocity as well as an interface to the game clock. `Actor`s have a large overhead, much of which is redundant for AI constructs that are contained within larger game entities like soldiers. For example, the physical location of a `Controller` is always the same as that of the soldier it directs, so storing it independently is unnecessary. However, a dependency on the added functionality of `Actor`s required their use for most of the AI classes, despite the `Object`'s smaller memory requirement.

`Pawn`s represent everything in the world that can be controlled by a player or AI module. All avatars in the game are represented by `Pawn`s; all other active parts of the world (guns, doors, etc) are `Actor`s. All `Pawn`s have a `Controller` (which tells them what to do) and an `Inventory` (which tells what they possess) in addition to various physical properties which define how they interact with the world (collision volume, physics information, eye-height, etc). A subclass of `Pawn` is used to represent all people in the Army Game Project, including soldiers.

`Controller`s are the subclass of `Actor` that governs how `Pawn`s actively interact with the environment. `Controller`s interface with the underlying code to do appropriate animations and update physical characteristics when so directed. `Controller`s also receive all perception events and handle them appropriately, which is significant for AI purposes.

`Controller`s are further subclassed into `HumanControllers` and `AIControllers`. `HumanControllers` take inputs from user devices (keyboard, mouse, joystick) to decide which actions to call; `AIControllers` have default no functionality but provide a basis for the control structures of a `Bot`, which is the `AIControllers` used for existing Unreal applications (deathmatch, capture the flag, and so forth). `Bots` are state machines as discussed in Chapter 2. UnrealScript™ requires that a `Bot` only be in one state at a time.

3. Limitations

There are a number of limitations to the existing system that make it inappropriate for the goals of this thesis. First, perception is a Boolean quantity; something is either perceived completely or not at all. Hearing does allow for a `loudness` parameter to be passed with the perception `event`, usually for presentation to a human player through the headphones or speakers. Sight is based on raytracing between the perceiver and the target, using several rays to ensure that partially occluded targets can still be seen.

Second, the control is almost completely reactive. Its decisions are almost completely determined by its immediate perceptual environment, with no planning and little memory of previous entities or events to guide it. When tracking an opponent beyond line of sight, for example, all it knows is the last place where they were seen.

Third, decisions are entirely state-dependent, and therefore nearly deterministic (in a certain state, the `Pawn` will decide to do a certain thing without fail). Each state consists of a long block of complex branching code controlling how the `Controller` acts and responds to perception events. There is no ability to adapt or alter its tactics in situations not anticipated by the designers. Non-determinism is only created by the interactions between multiple `Bots` and the asynchronous nature of interaction between the `Bots` and human players, along with a moderate dependence on random numbers for weapon accuracy and state selection.

Fourth, `Bots` are designed to be rational. Since the goal of most games is to give the players a challenge, it is not particularly meaningful to make the `Bots` choose “bad” goals or actions. Although they are hampered in some ways (giving them less than perfect accuracy, for example), they always choose the best state that the designers could anticipate for their situation. In a reality-based game where the `Bots` are designed to mimic real people, this is inappropriate; morale failures, confusion, and bad tactical choices need to be within the realm of possibility.

The design of the AGP AI module deals with each of these areas of deficiency in a way that allows for more 'human-like' behavior, including limited and imperfect perception, memory (and forgetting), inference, non-deterministic behavior, sharing of information, and cooperative goals.

C. AGP AGENT IMPLEMENTATION

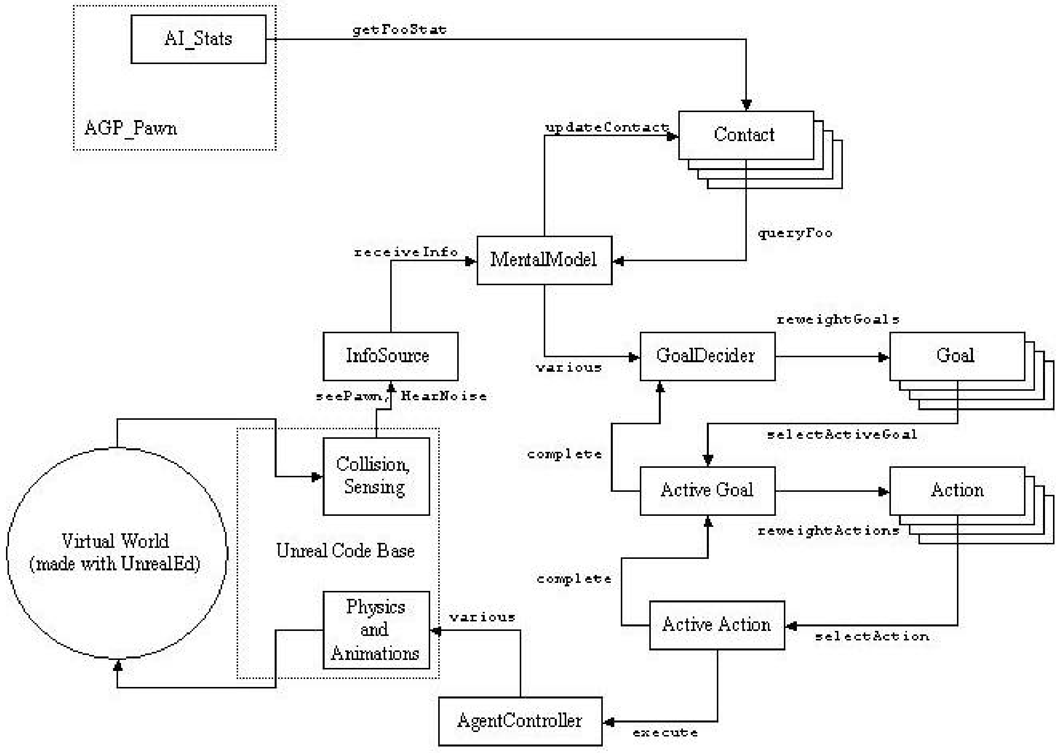

Figure 4. Information - Decision - Action Loop

In keeping with the guidelines described in Chapter III and the options and restrictions imposed by the structure of the Unreal™ system, the AGP AI module has the following components:

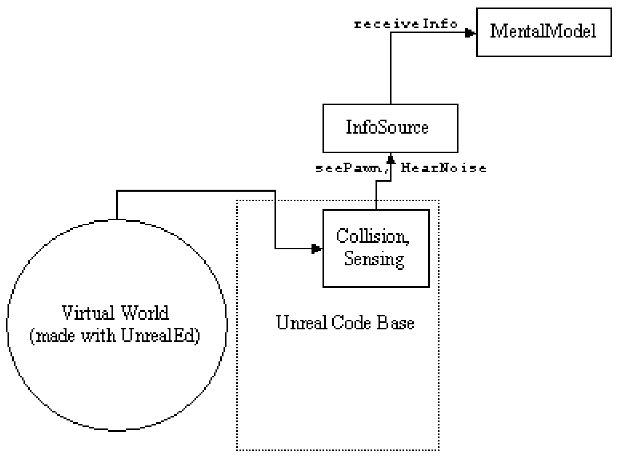

Figure 5. `InfoSource`

1. `Class InfoSource`

Since the `Controller` catches seePlayer, seeMonster, and hearNoise events, it was necessary to package these events and send them to my own `Class` for proper processing. With a slight modification to the `AgentController` already being used, all perception events were passed into the `AGP_Pawn`'s `InfoSource` along with the type of perception and its intensity (if appropriate). The `InfoSource` then called the `MentalModel`'s `receiveInfo` method to dump the relevant information into the mental model.

After some consideration and testing, it became apparent that it was appropriate to hard-wire certain reactions directly into certain stimuli. Actions that are done without thought, such as ducking when presented with a loud, unexpected noise, are inappropriate to handle through a goal-decision process, as they must be done quickly and automatically. Therefore, when the `InfoSource` registers a hearNoise event above a certain threshold (affected by both the loudness of the noise and the relative calmness of the environment), it will directly trigger the `AgentController` to duck or go prone. Once the `Pawn` is in a stressful situation, the noises of gunfire (for example) will no longer be loud enough to cross the higher threshold, allowing the agent to act normally.

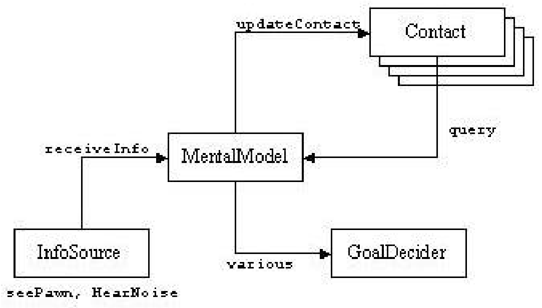

Figure 6. `MentalModel`

2. Class `MentalModel`

The `MentalModel` is the repository for all information about the environment known by a particular agent. It contains an array of `Contact`s (see below) that it updates when it receives perception calls from the `InfoSource` and queries when asked about one of them.

Optimally, each soldier should have its own `MentalModel`; however, this would require significant memory and processing time. Furthermore, since soldiers travel in units that are typically in sight and hearing of each other, there would be a large amount of replication; that is, more than one soldier would have essentially identical models of the world. Additionally, since these soldiers would be in close proximity for most of any given mission, there would be a large amount of traffic needed to communicate information between them. To solve these problems, soldiers in a squad share a `MentalModel`. Each soldier has their own `InfoSource` which links to the model, but as long as they all stay part of the same squad, they enjoy continuous and perfect contact with their fellows.

`MentalModel`s also have several utility methods to compute useful quantities about the environment. They may be called to assess how much threat a particular `Contact` presents to the caller, and the overall assessment of all threats combined. Conversely, it may also provide a calculation of how vulnerable a particular `Contact` is to the caller and therefore how attractive a target it makes.

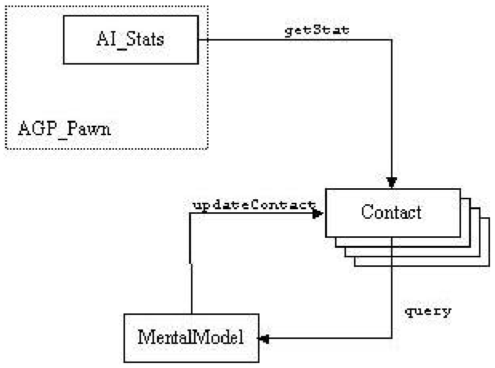

Figure 7. `Contact` and `AI_Stats`

3. `Class Contact`

A `Contact` is a representation of a single entity in the `MentalModel`. `Contact`s do not contain entity information themselves; instead they store a pointer to the `AI_Stats` (see below) of the entity in question and an abstract `infoLevel` that rates how much the caller knows about the `Contact` on a continuous scale from 0.0 to 1.0. The `infoLevel` is further divided into static and dynamic; `staticInfoLevel` does not decrease over time, and is used when accessing traits that are relatively unchangeable, like alignment (friend or foe) and entity type (infantry, civilian).

`DynamicInfoLevel`, on the other hand, rates the caller's knowledge of the entity's current characteristics that change over time: position, velocity, condition, current goal, and so forth. This decreases over time at a fixed rate. This method of degradation is a compromise between the losses of information due to actually forgetting what was seen and heard and those due to the actual parameters changing. The `MentalModel` continuously updates the `infoLevel`s as long as the `Contact` is in sight, so these do not come into effect until a wall or terrain feature occludes it. After this, the `MentalModel` still knows the location of the target, simulating its ability to infer or guess where someone will go right after they duck out of sight. After a certain amount of time, the `dynamicInfoLevel` degrades to zero and the `MentalModel` can access no information about the `Contact`'s dynamic qualities.

4. Class `AI_Stats`

The goal of the `AI_Stats` class was to create a single module to contain all AI-relevant information about any type of `Actor` in the virtual environment. To accomplish this, an `AI_Stats` does not have any predefined attributes. Instead, it holds arrays for information with floating point, enumerated, or vector values, and supports methods that allow new characteristics to be added, stored, and updated as needed. Thus, the same `AI_Stats` class can support an enemy soldier or an M16A2 rifle, even though their meaningful quantities are completely different.

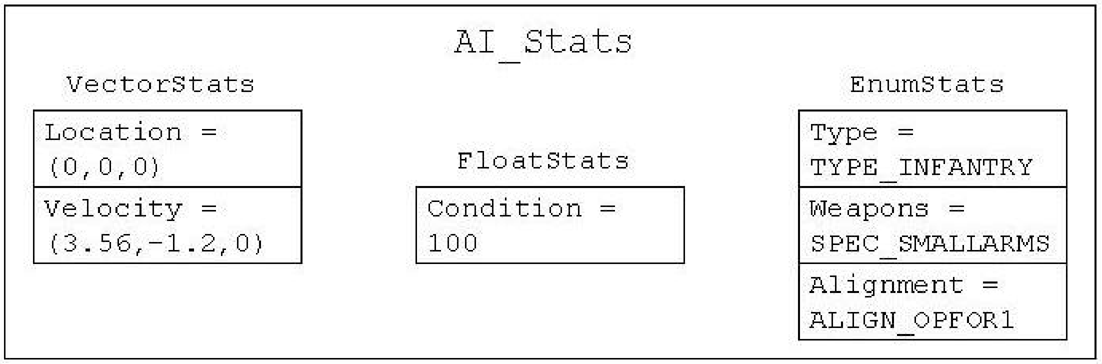

Figure 8. Sample `AI_Stats` for an Opposing Force Infantryman

Each characteristic is stored as either dynamic or static. This is initialized with the stat and indicates whether or not its value will change. Static quantities are assigned upon creation and left. Dynamic quantities are more problematic; in UnrealScript™ there is no way to create pointer to a primitive-typed value, nor is there a way to dynamically associate a function call with the stat to retrieve the source value elsewhere in the system. There are two ways to handle this problem. The first is to find all conditions under which the desired value changes and add a call to the `AI_Stats` to update its value as well. This is the best solution since it guarantees that the `AI_Stats` will give the correct value whenever it is called. For most variables, however, it is inappropriate or impossible, either because they are changed in many places in code or because the engine directly manipulates them, outside of script. For these stats it is necessary to set up a timer to update the `AI_Stats` value periodically. This is less efficient and uses up extra processing time, but is necessary for certain stats like location and velocity. Fortunately, most stats are either entirely internal to the AI, static, or both.

`AI_Stats` is further subclassed into `AI_ObjectiveStats`. The principal addition is that `AI_ObjectiveStats` are automatically placed as a `Contact` in all `MentalModels` when the game begins. This is to represent objects in the virtual environment whose location would be known to everyone, like buildings or bridges.

The `AI_Stats` interfaces with the `Contact` using the `infoLevel`s as described above. The `infoLevel` passed to the `AI_Stats` determines whether it returns the actual value desired, a predefined value indicating "unknown", or a misleading value. Currently, the code only supports a single spoof value, although future work might include multiple, dynamically assigned, or randomized spoofs. The inclusion of spoof values ensures that spurious values can be introduced into the decision process, and therefore lead to inappropriate decisions.

One shortcoming of this approach is that there is no persistence of information; with the `infoLevel` going up and down over time, it is quite possible to request the value of a stat twice in sequence and get two different return values. Fortunately, the boundary cases where this occurs should be both rare and transparent. `InfoLevel` will, in general, be increasing (with continuous perception events) or decreasing (through forgetfulness, and only for dynamic stats). Therefore, there should be only one conflict between function calls in any short period of time as the `infoLevel` crosses the threshold. Since the information is being pulled from other modules rather than pushed steadily from the `MentalModel`, it is unlikely that such an event would occur. Even if it did, it is not unlike suddenly coming to a conclusion after being initially unsure about what was seen. Since the returned values are not shifting back and forth rapidly, they should not lead to a thrashing situation in the goal decision process.

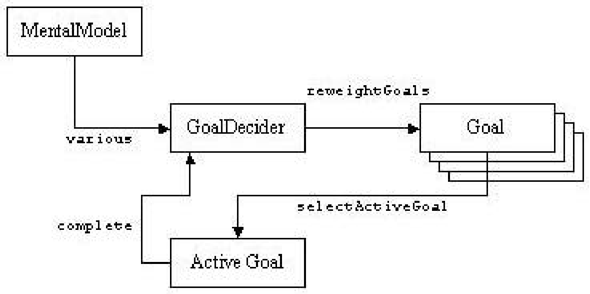

Figure 9. `GoalDecider` and Goals

5. `Class GoalDecider`

Although this class is the heart of the agent architecture, it consists of only a simple goal-delegating authority. When it is created, it has a number of `Goal`s added to it, representing the motivations that the agent will pursue in the virtual environment. As described below, each `Goal` must know how to weight itself based on the state of the virtual environment (as known through the `MentalModel`) and the state of the agent itself. It must also know how to execute itself and know when it is complete or has been interrupted. Since the `Goal`s contain all of this information, all the `GoalDecider` must do to reweight them is to ask each one their weight and pick the highest. Then, when the `Goal` is complete or interrupted, it asks the `GoalDecider` to reweight itself, starting the process over.

Of course the environment will produce situations where it is appropriate for the `GoalDecider` to reweight before it has finished with its current `Goal`. For this reason, the `MentalModel` forces a reweight for all of its members whenever a new `Contact` appears, or when a `Contact` reappears after being hidden for a period of time. Thus, the `GoalDecider` should never be caught executing an old `Goal` after it has gained new information.

6. `Class Goal`

`Goal`s are similar to the `GoalDecider` in structure. Each has a number of associated `Action`s, each of which can rate its effectiveness under the current circumstances. When activated, the `Goal` asks each of its `Action`s to weight themselves, and then executes the one with the highest weight.

As with the `GoalDecider`, when an `Action` is complete or is interrupted, it notifies the `Goal` so it can pass execution back to the `GoalDecider` for a reweight.

Not all agents need to know about all `Goal`s when the game begins. Because `Goal`s can be assembled and added to the `GoalDecider`'s list dynamically during play, a leader could know or learn of a mission and then pass it down to his subordinates as the results of some other event in the game. `Goal`s may also be generated entirely separate from any agent in the game and 'discovered' based on specific events (e.g. radio communications from HQ, or finding a piece of intelligence).

Table 3 shows the `Goal`s that are currently implemented, and their general use.

Goal	Description	Associated Actions

`GoalAttrit`	Move to and attack targets of opportunity.	`ActionAttack`
		`ActionFixJam`
		`ActionReload`
		`ActionMoveTo`
		`ActionAdjustInv`
		`ActionRecovery`
		`ActionIdle`

`GoalDefend`	Move towards specified objective and defend it from enemy forces. May be used either defensively or offensively, depending on whether the soldier is currently in control of the objective.	`ActionAttack`
		`ActionFixJam`
		`ActionReload`
		`ActionMoveTo`
		`ActionIdle`

`GoalImprove`	Improving personal state through by resting, reloading ammunition, unjamming weapons, or just looking around.	`ActionFixJam`
		`ActionReload`
		`ActionAdjustInv`
		`ActionRecovery`
		`ActionIdle`

`GoalSurvive`	Take cover, avoid or run away from threats.	`ActionTakeCover`
		`ActionFlee`
		`ActionFixJam`
		`ActionReload`
		`ActionIdle`

Table 3. Implemented Goals

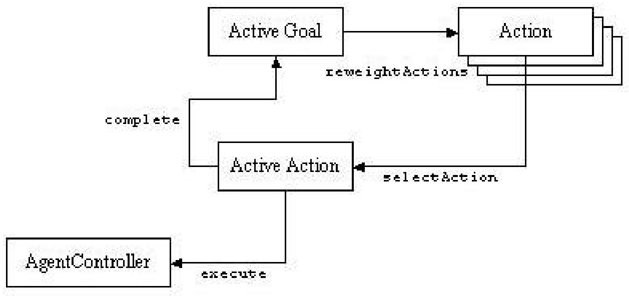

Figure 10. `Action`s

7. `Class Action`

`Action`s interface with the existing finite state machine structure in the `AgentController`. All `Actions` know how to weight themselves, as with `Goal`s, and what state to activate in the `AgentController` when they are executed. `Action`s also register themselves with the `AgentController` when they become active so it knows where to return the notification of completion or interruption. The `Action`s then notify their parent `Goal`.

A listing of the currently implemented `Action`s is given below.

Action	Description
`ActionAdjustInv`	Switch to a more effective weapon.
`ActionAttack`	Turn towards enemy and fire weapon.
`ActionFixJam`	Unjam a jammed weapon.
`ActionFlee`	Run to the nearby location that combines closeness, cover and maximum distance from the current threat.
`ActionIdle`	Stand in place and look around for enemies.
`ActionRecovery`	Move to and retrieve weapons and ammunition.
`ActionReload`	Reload an empty weapon.
`ActionTakeCover`	Drop to a crouched or prone state.

Table 4. Implemented `Actions`

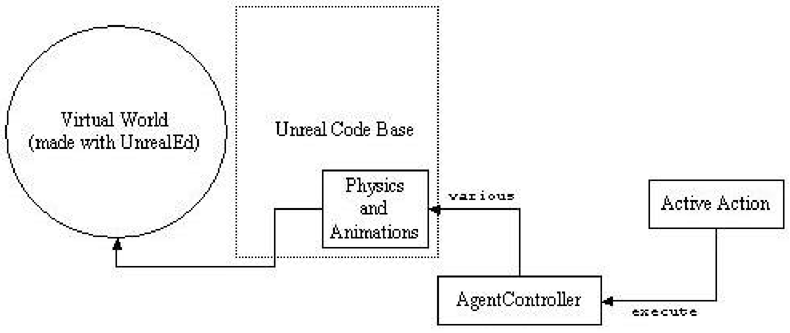

Figure 11. `AgentController`

8. Class AgentController

The `AgentController` is very similar to the `Bot` class in the default Unreal™ AI, as described in the first part of this chapter. In the AGP, all `AgentController` states directly correspond to one of the `Actions`. When executed, the `Action` tells the `AgentController` to enter the appropriate state and begin execution. The state code determines what to do and calls the appropriate functions to actually move, turn, fire weapons, trigger animations, and all other effects within the virtual environment. If the `AgentController` enters another state without properly completing the one it is currently in, it returns a message to the original `Action` that it was interrupted; otherwise, it completes the state code, tells the `Action` that it was completed successfully, and waits for the next jump to a new state.

The implementation within each state is highly variable. Some `Actions` are fairly atomic. For example, reloading the weapon is a single function call to the engine, and is only applicable when the weapon is empty and a `Goal` that cares about the weapon state is active. At the other extreme, the `MovementTo` state makes numerous decisions about choosing a path to follow, turning, and triggering appropriate animations. Firing a weapon is also complicating, requiring a check for uninterrupted line of sight, range to target, and weapon status before turning to face the target. When all conditions are met, it can then tell the weapon to trigger the animations and physics checks.

D. CONCLUSION

Overall, the goal of the agent-system AI is to enable the soldiers to perceive their environment, remember and process its vital aspects, and make weighted decisions about what to do in real time. Additionally, soldiers need to share information and have some facility to be misled by imperfect or incomplete information. At the same time, it is desirable to reuse as much code as possible from the existing `Bot` implementation.

The implementation described above fulfills all of these goals; some areas, like the goal decision process, are closely in accordance with the guidelines of designing an agent system. Others, such as the shared mental model, are concessions made to the realities of real-time processing and the need for acceptable decision and graphics refresh rate. Despite these compromises, the effect appears to be indistinguishable from individual soldiers having their own mental models, at least in the test cases conducted in the lab so far. Chapter V will discuss the testing used to determine whether the design goals were met for the intended audience.

V. TESTING AND ANALYSIS

A. INTRODUCTION

This chapter describes the subject testing of the AI system developed in this thesis. First, it discusses the conceptual framework and goals of the testing process, and then the test protocol and the results obtained. The survey data are presented concisely to demonstrate the experiment's outcome. Finally, the conclusion discusses some avenues of improvement based on the survey results.

B. TESTING PHILOSOPHY

The overall design goal of this thesis is to produce behavior more consistent with a hierarchical, cooperative military organization than the game AI used in commercial and academic systems. Underpinning this goal is a need for general improvement in the fidelity with which human behavior is modeled in such systems. If the individual soldiers do not act like real human beings (as opposed to human players in an FPS game), there is little hope that a group of them will behave realistically, much less like a time of soldiers.

There are a number of factors that have some bearing on the test subjects' ability to gauge humanlike and soldierlike behavior, and these will be taken into consideration for the experimental design.

1. Subject Game Experience

Familiarity with FPS games will most likely have a significant impact on the subject's perception of the agents' behavior. Those with the most experience with this sort of game will need to expend less attentional resources on the basics of game interaction and thus should be able to pick up finer details of what is going on in the game itself. Additionally, familiarity with traditional FPS games will also condition their expectations of AI behavior, thus making any differences in the AGP system more apparent.

2. Subject Military Experience

Presumably, the subject's degree of military experience will determine the level of fidelity necessary to convince them that the agents are thinking and acting in an appropriate way. A given user might have any degree of experience, from complete novices to senior infantry officers with combat experience. It is likely that inexperienced subjects will miss behavioral inaccuracies that a veteran would find jarring and unnatural.

3. Game Presentation

Since the AGP is designed to be run on standard PCs, its interface and presentation are limited to a keyboard and monitor. Monitors have two main limitations that may affect the subjects' ability to pick out the agent behaviors. First, their field of view is fairly limited, typically around 30-45 degrees total viewable arc in the virtual world. This makes it difficult to see a full picture of the world and specifically the soldiers in it. When the player is approaching a group from a distance, they may be able to see all of the enemy agents, but once they are surrounded or involved in the action, they may miss important events that happen within their (human) perceptual envelope, but outside that of the monitor.

Similarly, the speaker systems presenting game sounds are usually fixed to the monitor or on either side. Audio spatialization is essentially non-existent, and thus the players' ability to respond properly to sound events is hindered. It is not clear whether this limitation makes proper assessment of the agents more difficult.

4. Game Content

The trial kept to games with the most realistic content possible: only human characters, and real modern weapons and uniforms. The depiction of game characters was as realistic as possible, with appropriate animations, blending, shading, and sound effects to generate a proper environment. This aided in evaluating the AI, since it was less likely that players would be distracted or misled by the novelty of a fantastic or science-fiction setting. Additionally, the higher the presentational fidelity, the more an observer expects other aspects, like behavior, to be correct as well. Since correct behavior is the goal of this thesis, this was a good expectation for the subjects to have.

Nevertheless, there are a finite number of discrete states and actions that a game character can perform, and therefore some desired actions may be impossible. For example, it may be reasonable in real life that a soldier crouch down behind a wall so that just their weapon and the top of their head are over the top, to maximize cover while retaining the ability to see and shoot. In the game, the wall would have to be exactly the proper height in order to support this; if it were taller than the standard crouch height, it would occlude the weapon and the eyes; if it were too short, more of the soldier would be visible.

Situations like this present difficulties in two ways. First, the players may be distracted by their inability to perform common-sense actions due to the constraints of the game and interface. This distraction takes attention away from the true content of the game, which may in turn hinder a proper assessment. Second, when the players see the AI-controlled characters taking cover behind walls (to use the example above), they believe that the soldiers are doing something incorrect rather than perceiving the limitations of the engine's ability to represent physical actions and positions. As such, they might ascribe seemingly odd or illogical behavior to the AI rather than to the underlying engine that governs its interactions with the virtual environment.

5. Environment Design

In addition to the functionality of the game engine itself, the design of the environment is vital to the ability to demonstrate behavioral qualities of the agents. The first concern is that the agents' actions be apparent to the characters through as much of the trial as possible, without making it too easy to kill them or die. This argues for an open space (as opposed to one set indoors) with prevalent hard cover so that observation is possible without being shot. Nevertheless, there should be significant risk and room to maneuver so the agents (and the player) can explore various tactical situations and evoke the full range of behavior from the agents.

6. Control Game

In order to demonstrate a difference from existing, rules-based AI systems, a comparison is required. As such, the subjects need to use both the AGP and another system side-by-side and compare their observations of the agents' behaviors against those of the AI bots in the other system.

In order for the comparison to be meaningful, several experimental design goals must be met. The environments must be as similar as possible so the range of interactions and tactical situations is the same, but at the same time different enough that play in one game does not introduce learning effects in the second. The physics and animation detail should be of the same quality so that presentational differences in the virtual environment do not detract from the comparison of broader behaviors. The game's range of controls should be similar so that the players' ease in using the interface does not affect their experience differently between the two games. The same subjects should play both games so there is a direct comparison between them. The games should be short enough that learning has a minimal affect on the player's ability to change their ability to function in the second system based on their knowledge of the first, but long enough that they have a chance to interact fully with the system and the agents or AI controlling the opposing force. Finally, the order in which the two games are played should be mixed so that effects of the order played can be discerned during analysis.

7. Survey

The observed differences in behavior between the traditional AI and the agent system must be subjectively assessed. Since the goal is 'proper' behavior rather than 'optimal' behavior, there is no objective method that can be used to gauge success. Rather, each subject must decide how closely their perceptions of the soldiers' behaviors matches what they would consider appropriate. Then it is up to the experimenter to correlate the subjective opinions of the individual subjects and their own knowledge and experience.

One approach is to create a list including both useful questions about their perception of the behaviors and misleading questions about other aspects of the game to prevent any conscious or subconscious effort to aid (or hinder) the goals of the experiment. Another is to ask pointed questions about the agent behaviors and comparisons between them and those in the commercial product with the hope that the subjects would respond with a greater depth of useful information. The experimenter might even prime the subjects beforehand to look for and attempt to evoke a wider range of tactical situations to push the limits of the AI. As suggested previously, though, the knowledge of the goals of the experiment are likely to taint the subjects' responses on the surveys. Even with the best intentions, a subject might err one way or another in an effort to fulfill the experimenter's expectations for the thesis.

C. SUBJECT POOL

The choice of a subject pool is important for obvious reasons. If it is too narrow, there is the risk of missing vital knowledge or skills that could evoke new and different reactions. If it is too broad, then the number of factors affecting the evaluation could easily outstrip the number of subjects, making analysis virtually impossible.

Two pools of subjects were available for this particular testing period. Each deserves some specific discussion.

1. NPS Student Body

The student body of the Naval Postgraduate School is a clear candidate for subjects. Nearly all students at the school are mid-grade and senior active duty military personnel, ensuring that they have at least some experience with military matters. For those who are not Army or Marine Corps infantry officers, however, this may not appreciably increase their ability to evaluate infantry combat behavior. Additionally, this group has the advantage of being located on campus, and would likely be highly available.

2. High School AJROTC

The other possible subject pool was the Army Junior Reserve Officer Training Corps (AJROTC) unit at the local high school. While the students have less experience with military affairs outside of their weekly class, they do have interest in the subject matter, which is important for promoting attention and focus on the evaluation. The AJROTC unit was chosen primarily because they were available regularly and in large groups due to their drill schedule, thus making it possible to work through many individuals in a short period of time without extensive scheduling.

D. EXPERIMENT PROTOCOL

The purpose of the experiment was to establish that an agent system is at least equivalent to traditional AI for producing believable, soldier-like behaviors in a FPS game with a modern military setting. To this end, the experiment attempted to compare two similar games in such a way as to make the differences in behavior as apparent as possible.

The experiment was conducted on March 11th, 2002, with subjects drawn from the Army JROTC unit at Seaside High School, Seaside, California. A total of 16 students completed the experiment and filled out the survey form.

1. Preparation

First, an appropriate level was designed for the AGP, with a small number of AI-controlled soldiers, a small structure, lots of natural and artificial cover, and open lines-of-sight.

Return to Castle Wolfenstein™ was chosen to be the control case game. While it is a fantasy game, it has the capability of being completely realistic, at least within the constraints of World War II technology and uniforms. Additionally, it is military-oriented, reducing the number of visual dissimilarities between it and the AGP. Basic user interface and avatar actions and motions were similar enough between the games that there was be no problem playing one game after another, especially with a familiarization period before each game.

A similar level was generated for Castle Wolfenstein. Each program was loaded on a computer, and both were transported to Seaside High School and installed in the AJROTC administration office. This location was physically separate from the classroom to create as much isolation as possible for the test subjects and prevent interference with the other students.

2. Introduction

At the beginning of each trial, pairs of subjects were brought to an isolated area containing the testing platforms. Each was given a short verbal brief on the testing procedure and signed the appropriate human subject release forms. The students were told that they were assisting with a thesis experiment to compare a commercial game to a game under-development by the Naval Postgraduate School.

Once briefed, each of the subjects was sent to one of the computers, either the control case or the test case. After completing one game, they switched places.

3. Control Case

The subjects were given a short introduction to the game. They were given a note card with the standard keyboard commands and placed in an empty level to test out the controls and become familiar with the interface. After two minutes of familiarization, the actual test level was loaded and they played in it for eight minutes, under the observation of the experimenter to answer any questions or deal with any software issues. The game was restarted when the character died, ran out of ammunition, or killed all of their opponents and ended the level. At the end of the test period, the simulation was stopped.

4. Test Case

The AGP software was started on an empty level to test the interface in the same manner as given for the control case. After two minutes, the test level was loaded and they were allowed to play for eight minutes, again under observation. The level was reloaded any time the character died, ran out of ammunition, or completed the level. The game was stopped at the end of the testing period.

5. Survey

After completing both games, the subjects were taken aside and given a survey form (see Appendix B). They were allowed whatever time necessary to complete the form, while the next subjects were starting the experiment. Subjects filled out the survey form individually and anonymously. If the subjects had questions about the survey, they were able to ask the experimenter, but no additional guidance was given. When the survey was complete, the forms were returned to the experimenter and the subjects returned to their class.

E. RESULTS

This section will discuss explicitly the contents of the survey filled out by the subjects. Both the raw data and some reasonable associations will be pursued with an eye towards their positive or negative consequences.

1. Demographics

Subjects were asked to several pieces of personal information to help the experimenter identify trends and commonalities in the survey responses. The subjects were asked for the age and gender, as well as the frequency with which they played FPS computer games and the likelihood that they would consider a career in the army. These questions were included to demonstrate that the subject pool represented the target audience of the AGP. None of the factors polled were judged to have a definitive effect on a given subject's ability to assess the realism of the AI's behaviors. FPS experience, in particular, could be problematic since it could be interpreted as a positive or negative factor. It would indicate a superior ability to operate the interface and perform in the virtual environment, and therefore the ability to evoke and observe AI behaviors, but it might also taint the ability to compare the behavior to reality instead of similar FPS games. This experiment will not draw any conclusions from these factors, relying instead on the ratings of the games by the population as a whole.

2. Assessment of the AI

After completing both games, each subject graded the realism of each game in five categories, which are described on the survey form and in Table 5, below.

Category	Description
Response	Reactions to the player's moveme nt and actions; range of sight and hearing; ability to track the character when out of direct line of sight.
Appropriateness	Choice of actions in response to the current situation; good choice of tactical decisions based on available information and physical condition.
Self-Preservation Behaviors	Actions taken to evade, take cover, and avoid the player when they need to.
Aggressive Behaviors	Actions taken to track, pursue, and attack the player.
Other Behaviors	Actions while wandering around, ignoring the player, or unaware of the player.

Table 5. Survey Judgment Parameters

Subjects rated each category for each game, using discrete assessments of *Totally Unrealistic*, *Mostly Unrealistic*, *Moderately Realistic*, *Very Realistic*, or *Totally Realistic*. Because the marks for the two games were placed side-by-side on the survey form, the subjects certainly marked them both in reference to their personal standard of realistic behavior and against each other. Therefore, the mark given to the AGP is relevant by itself as well as in comparison to the mark given to Return to Castle Wolfenstein™.

3. General Results

Tables 6 through 11 below show the overall results of the survey in each of the categories given above. "#" indicates the number of subjects giving a particular response, and "%" indicates the percentage of the subject pool.

The subjects answered the same questions for both the AGP and Wolfenstein™ soldier behaviors. However, since the purpose of this experiment was to constructively compare the two, the raw results for Wolfenstein™ are unimportant except for how they relate to the AGP's results. Since the ratings are subjective, the statistics will only summarize which game was considered more realistic in each category. More detailed may be extracted from the survey results themselves, in Appendix B.

Age	**15**	**16**	**17**	**18**
#	4	5	4	3
%	25%	31%	25%	19%

Table 6. Subject Age Distribution

Gender	**Male**	**Female**
#	12	4
%	75%	25%

Table 7. Subject Gender Distribution

Gameplay frequency	**Never**	**Once or Twice**	**Monthly**	**Weekly**	**Daily**
#	3	3	3	5	2
%	19%	19%	19%	31%	12%

Table 8. Subject FPS Gaming Frequency

Likelihood of Army Career	**No**	**Unlikely**	**Possible**	**Likely**	**Definitely**	**?***
#	0	3	4	6	2	1*
%	0%	19%	25%	38%	12%	6%

* one subject reported their interest as "?" instead of using one of the categories provided

Table 9. Subject Army Career Likelihood

	Rating				
AGP AI Category	**Totally Unrealistic**	**Mostly Unrealistic**	**Moderately Realistic**	**Very Realistic**	**Totally Realistic**
Response	0 0%	1 6%	2 12%	9 56%	4 25%
Appropriateness	0 0%	1 6%	3 19%	7 44%	5 31%
Self-Preservation	0 0%	0 0%	4 25%	5 31%	7 44%
Aggressive	1 6%	1 6%	4 25%	4 25%	6 38%
Other	1 6%	1 6%	3 19%	10 63%	1 6%

Table 10. Subjective Ratings of AGP Realism

	Rating (compared to Wolfenstein ä)		
AGP AI Category	**Worse**	**Equal**	**Better**
Response	4 25%	3 19%	9 56%
Appropriateness	3 19%	9 56%	4 25%
Self-Preservation	1 6%	8 50%	7 44%
Aggressive	4 25%	6 38%	6 38%
Other	3 19%	8 50%	5 31%

Table 11. Relative Ratings of AGP Realism

Although a sample size of 16 is insufficient for any rigorous statistical purposes, there are a number of trends which appear positive. In absolute terms, the AI performed well; an average of 72% of the subjects found the AI to be *Very Realistic* or *Totally Realistic* overall. However, this is based on unreferenced opinion, and is therefore not particularly meaningful. More important for the purposes of this experiment is that it compared very well with a traditional AI technique when rated for realism. An average of 39% of the subjects felt that it had more realistic behaviors than Return to Wolfenstein™, and only 19% felt that it was less realistic. The remaining 42% felt that the behaviors were of comparable realism. Considering that the experiment is comparing a production game with one that is still under development, parity should also be considered success. Further development will only improve the abilities of the AGP AI.

The survey also had a space for subjects to make any additional comments they wished. Most of the subjects did give additional information, although it rarely mentioned the observed behavior of the soldiers. Nearly all of the comments were positive mention of the AGP's graphics, gameplay, and weapon effects modeling, although there were some comments on physics irregularities and other observed bugs. Several subjects also made positive comments about the test case game. There were two AI-relevant comments: first, that the AI soldiers ran away too often, and second, that it seemed especially realistic when an AI soldier ran up to check the player's body after they had been shot. These comments suggest further avenues for improvement and analysis for the AI as well as the AGP as a whole.

F. CONCLUSION

Although the results of this experiment cannot be considered proof in any statistical sense, they do show some trends. On the whole, the subjects considered the AGP AI code to be at least equivalent to that of a production FPS game in producing realistic soldier behavior. As such, the code can cautiously be considered to have achieved its goals, at least at a basic level. The results of the experiment also suggest likely avenues for improvement to the system. For example, the subjects considered the soldiers' incidental behaviors (described as "Other" above) to be the least appropriate, only averaging halfway between *Somewhat Realistic* and *Mostly Realistic*. This suggests that soldiers who do nothing but attack, defend, and run away are less realistic than those who wander, converse, look around, and engage in similar actions that are not directed at a combat-oriented goal.

Similarly, the comparative ratings indicate which areas of the code are more or less in need of further development and tweaking. For example, 44% of the subjects considered the self-preservation behaviors to be more realistic than the test game, as opposed to only 6% who thought they were less realistic. This suggests the weighting scheme and actions for `GoalSurvive` are reasonable, or at least pointed in the right direction. On the other side, only 25% of the subjects thought the AGP soldiers chose more appropriate actions, and 19% thought they chose less appropriate actions. This should lead to further tweaking of the `InfoSource`, `MentalModel`, and `GoalDecider` code to improve the fidelity of the information transfer, inference, and decision-making.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND FUTURE WORK

A. INTRODUCTION

This thesis has demonstrated that an agent-based system can effectively simulate soldier behaviors in a dynamic 3D virtual environment as typified by an FPS game. Producing this realism required a number of innovations in FPS AI control, including imperfect perception, mental modeling, an agent-based goal decision process, and a series of informational feedback loops to allow the system to respond to a changing environment. Furthermore, testing suggests that an agent-based approach to modeling soldier behavior is superior to traditional rules-based AI in many ways.

B. INFORMATION FLOW

Realistic behavior depends on realistic modeling of the means for gathering and storing information about the environment. Although the perception implementation used the default Unreal events to determine when an entity was seen, modifying that Boolean event into a more fluid concept of `infoLevel` allowed the soldiers to slowly increase their knowledge and change their behaviors as more information became available. Also, the possibility of spoofing leads to the possibility of making poor tactical decisions, which is a desirable for representing how many decisions are actually made in combat and other stressful situations.

The `MentalModel`'s slow degradation of `infoLevels` gives the soldiers the ability to make inferences about the location and properties of the other soldiers, since they will continue to "know" about them even after they have left the perceptual field. While this may eventually produce unrealistic results, it bypasses the extremely difficult problem of making actual inferences based on perceptual data, which could be a thesis unto itself.

C. DECISION METHODS

The heart of an agent system is the goal decision engine. This implementation follows the broad design of an agent system, using a number of competing goals each being compared with an heuristic weighting function to determine which is the most applicable. What it is missing is a feedback loop to alter the goal weights based on the outcome of their actions. This feature is used in many agent systems to allow an entity to optimize its behavior and adapt to a changing environment over time. However, since a poorly chosen `Goal` or Action would typically result in death in the dangerous AGP environment, this adaptation was not deemed to be worthwhile. Even under the best circumstances, agents that learned to improve themselves would be lost when a given game level ended. Therefore, adjustments of the goal weighting heuristics were done by the designer rather than by the agent system itself. Lengthy testing seems to have produced reasonable goal-weighting schemes for those that are currently implemented.

D. FUTURE WORK

As with any thesis, the scope of the work to be accomplished narrowed dramatically over the course of the code development and writing. Many features and elements of the initial conception have necessarily been delayed in order to present a complete, tested, and fully functional code base. The following sections describe some of these avenues for future work, as well as some thoughts on implementation and the benefits they might impart to the AGP.

1. Testing

There are many ways in which the code could be tested beyond the limited trial noted in Chapter V. Since one of the goals of agent-oriented programming is to increase the intuitive nature of the code, it would be significant to survey code developers after significant work with both traditional AI and this agent system to determine whether this goal has been met. If not, further work or restructuring of the code might be required to improve the system's usefulness to future AGP developers.

Additionally, further testing of the code itself could be useful. Separating the four components and implementing them one at a time and in all combinations could demonstrate which modules are the most significant for producing realistic behavior. Based on the results of such an experiment, the designers could focus on improving the performance and fidelity of the most significant modules, resulting in an overall savings in time and effort.

2. Code Optimization

All of the AGP AI code is written in UnrealScript™, as described in Chapter IV. UnrealScript™ was chosen because it provided an intuitive way to develop the structure of the AI and made it simpler to organize the code modules for the system. The product of this choice is a functional proof-of-concept, but the relative slowness of script interpretation will eventually cause processor overload, especially as the number of soldiers, size of game environment, and length of mission all increase. Many of the functions and classes discussed in Chapter IV could be made `native`, moving their implementation into C++. The speed advantages of compiled versus interpreted code would greatly extend the maximum game complexity permitted.

This would require large amounts of code be rewritten, since the most likely candidates (`MentalModel` and `Contact`) are fairly lengthy. Nevertheless, the structure of C++ and UnrealScript™ are reasonably similar, so translation from one format to another would be tedious but not difficult. Furthermore, because of the flexibility of C++ the code could be optimized in ways not possible in UnrealScript™ in addition to receiving the advantages of being compiled.

Such efforts would benefit from testing with increasingly complex levels, possibly populating them entirely with AI-controlled entities. This could reveal the relationship between the numbers of various game environment constructs and the overall speed of the simulation, establishing an upper bound on the complexity allowed. This could be of great value to the design of future environments with the AGP.

3. AI-Sensitive Objects and Locations

Currently, only `AGP_Pawn`s and AGP_Objectives have functional `AI_Stats` and can therefore be recognized and stored in the `MentalModel` as `Contact`s. While important, there are many other aspects of the environment that would be useful to the AI. Since soldiers currently are unable to perceive anything without an `AI_Stats`, a greater diversity of AI-sensitive objects would certainly lead to more realistic behavior. For example, a mission focused on defending a vital bridge from the enemy would require the enemy AI to know of the bridge in advance and recognize its vulnerability to certain weapons, all of which would be encoded in its `AI_Stats.`

Instantiating an `AI_Stats` in chosen classes of objects would be very simple. New classes of objects in the environment might require new `Goals` and `Action`s to deal with them, or modifications to existing `Goals` and `Action`s (especially their `evaluate` functions). Thorough testing would show the degree to which the new responses changed the overall soldier behavior, and whether this improved or degraded its realism.

4. Class `GoalOrders`

Soldiers all begin with a set of instinctive goals and the overall mission goals in a given game environment. Additionally, all soldiers have an equal and independent means for deciding on their actions in their own personal situation. Fire team leaders and squad leaders are only special in that their fire team or squad explicitly protects them.

What may be useful is for soldier leaders (already distinct from followers in the code) to have an additional range of `Goals` available to them for the purpose of directing the soldiers under their command. These `Goals` would be constructed almost like those already in the system, but would not cause any direct actions when executed. Instead, they would temporarily add a new `Goal` to the `GoalDecider` of each subordinate soldier. Until the leader rescinded this `Goal` or gave a new one, it would be evaluated along with the rest of the `Goals` on an equal basis, thus temporarily expanding the soldiers' range of possible behaviors.

Such `Goals` would also benefit from an heuristic to determine the number and quality of troops required for successful accomplishment. Then, the leader could pass the order along to the desired number of troops (if less than the total size of his command) and leave the rest to accomplish some other goal or act instinctively. This heuristic could also benefit from a feedback mechanism whereby the leader could perceive how effective his force allocation was at completing the missions, moving troops from one job to another as they became more or less difficult or dangerous.

5. Aggregation and Disaggregation

A natural extension of giving different orders to two or more parts of a command would be allowing soldiers to break off of one `MentalModel` and form their own. This could be valuable for scouting missions, allowing single soldiers or small groups to travel away from their leader, establish new `Contact`s, and return to the parent unit to recombine and share their information.

Once the problem of choosing when to disaggregate is solved (as discussed in the previous section), it is relatively straightforward to actually implement it. As the soldiers physically separate from their parent unit, the leader's `MentalModel` would be copied and attached to the newly-chosen leader of the disaggregated unit; concurrently, the soldiers staying behind, already represented as "perfect" `Contact`s in the shared `MentalModel`, would be degraded to normal `Contact`s as the detachment moved away and began to "forget" where their fellow soldiers were deployed. From then on, the two would be distinct and not share information, so testing would be required to determine the appropriate conditions for actually breaking up the `MentalModel`.

Upon their return to the parent unit, the two groups and their `MentalModel`s could easily be rejoined. Administratively, all of the detached soldiers would return to be subordinates of the overall leader; the leader created for the detachment would lose his special status and become and ordinary soldier again. The newly combined `MentalModel` would have all of the `Contact`s contained in both of those being aggregated, choosing the best `infoLevel` available. Further work could allow less correct information to override more correct information, as the data is conditioned both for correctness (as with `infoLevel`) and the holder's confidence in that data.

Some difficulties would have to be overcome. The designer would have to decide whether detachments from two different fire teams could aggregate in this fashion to share information. Although more complicated, it may be necessary to produce realistic behaviors in small, cooperating groups from different commands.

The designer would also have to address the subject of inadvertent disaggregation. If a member of the group is unable to stay in physical proximity to his squad mates (due to injuries or blocking terrain, for example), his AI should force him to disaggregate until he can find his way back.

6. Psychology

In the currently implementation, all soldiers are copies of each other. Aside from some randomness in the generation of their default psychology (included in the Unreal system but not currently used by the AGP), all soldiers will tend to act in the same way under the same conditions. Although this is desirable to a certain extent, it would probably assist realism if some soldiers were more or less brave, aggressive, defensive, or lazy than the norm. These psychological factors could be easily integrated into the `Goal` or `Action` evaluation functions, modifying the chances that trait-appropriate `Goals` would be chosen. For example he bravery trait might influence `GoalSurvive` (negatively) and `GoalAttrit` (positively), whereas a laziness trait might increase the weight of all evaluations of `ActionIdle`, regardless of the `Goal` it was associated with.

Testing would be required to determine a “safe” range for each trait, so when soldiers are generated they do not have unrealistically high or low modifiers to their `Goal` evaluations. A soldier with a bravery of 0.0 and a laziness of 1.0 would immediately stand out because of their tendency to stand still whenever threatened; this might be useful for representing civilians but would be inappropriate for well-trained soldiers.

7. Friendly AI

The code currently supports soldiers from U.S. forces, generic opposing forces (OpFor), and generic neutral forces, as well as non-combatants. However, there has been no testing of levels populated with AI soldiers from more than one side. Furthermore, only enemy soldiers have been given AI for the thesis experiment. This choice was natural, since OpFor don’t have to share information with a human player. Thus, all information can be passed through function calls and the shared `MentalModel` without needing a physical or animation model of information transfer.

In order to integrate a human player into a squad of AI-controlled soldiers would require a number of modifications to the existing code. First, since the player is not part of the soldiers’ `MentalModel`, the designer would have to devise a scheme for graphically (or audibly) presenting `Contact` information and passing it on the player in an intuitive and useable fashion. Since players already have a shortage of available screen space, such information might be better displayed through icons or special shading and lighting in the screen depiction of the environment. However, such unnatural additions to the visual field would likely detract from the player's sense of immersion and might not even be useful as a decision aid if the `MentalModel` grows too large and cumbersome.

Secondly, the human player would have to have some way to determine the `Goals` of his squad mates, especially those of his squad leader. A player could easily become confused or separated from his squad without some idea of what they are doing and where they are going. In general, these `Goals` could be represented with graphics or sounds. Unrealistic graphics degrade the visual fidelity of the game, as discussed above. Sounds, especially speech, are much more appropriate to this sort of information. However, in order to sound realistic the AI would need at least one sound, word or phrase to play for each `Goal` available. This would increase the design overhead because it requires specialized code, hardware, and personnel.

Finally, all friendly AI would need some way to deal with the player. A number of the utility functions in `Contact` require knowledge of the target's current `Goal`. Although this can be directly perceived between AI soldiers by simple function calls, the AI has no way to extract a `Goal` from a human player. The compromise currently in the code is to rate human players as the maximum threat possible at all times. Overcoming this shortfall would require a way to evaluate the player's actions to determine their current `Goal` in terms of the `Goals` known to the AI. This might be as simple as sampling direction of movement and weapon discharge, but not likely. Because each `Action` can be called by an arbitrary number of `Goals`, it becomes ambiguous what `Goal` motivated a particular `Action`. Even at that, the process would almost certainly generate errors, since human behavior is much more continuous and dynamic than that of an AI with discrete and countable `Goals` and `Actions`. The field of "mind reading" to determine motivations is still a topic of much research in human psychology (Baron-Cohen, 1995); teaching an agent to do something similar, even in a narrowly scoped domain like the AGP, would be a daunting task.

E. CONCLUSION

Significant room is left to improve, expand, and optimize the AI code for the AGP. Nevertheless, the system as presented is a functional behavior generator for members of small groups of loosely-joined, self-motivated individuals in the virtual environment. It demonstrates that the finite state machine implementation of a traditional game AI is not the only alternative when attempting to generate humanlike behaviors.

APPENDIX A

A. INTRODUCTION

The source code involved in the AGP AI module is lengthy and divided between numerous files in several projects. Including it here in a meaningful form would have been nearly impossible, especially since it is highly dependent on the underlying Unreal™ code which is proprietary and therefore not subject to public release.

Nevertheless, portions of the code generated for this thesis are publicly available through the MOVES Institute. Please contact the AGP Project Manager, Dr. Michael Capps at capps@armygame.com or capps@nps.navy.mil.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B

A. INTRODUCTION

This appendix shows the forms used for the experiment in Chapter V and the explicit results of the survey. It is organized as follows:

- Participant Consent Form
- Minimal Risk Consent Statement
- Privacy Act Statement
- Warfare Game Realism Survey Questionnaire
- Summary of Survey Results

PARTICIPANT CONSENT FORM

1. **Introduction.** You are invited to participate in an experiment to compare a commercial computer game with a new game under development at the Naval Postgraduate School. You will play a scenario in the game Return to Castle Wolfenstein™ and one in the developing game, and compare the two in a short survey.

2. **Background Information.** Data is being collected by the Naval Postgraduate School's Modeling, Virtual Environments and Simulations department to provide feedback for behavioral and artificial intelligence design in dynamic virtual environments.

3. **Procedures.** If you agree to participate in this study, the researcher will explain all required tasks in detail. You will use the mouse and direction arrows to play a scenario in each of the games. The intent is for you to play the game to the best of your ability. The entire experiment will take approximately 25 minutes.

4. **Risks and Benefits.** The research presents the same risk as using a computer. If this is a problem, please inform the researcher before beginning. The benefits to the participants will be to contribute to current research in advancing behavioral design in dynamic virtual environments.

5. **Compensation.** No tangible reward will be given. If desired, a copy of the results will be available to you at the conclusion of the experiment.

6. **Confidentiality.** The records of this study will be kept confidential. No information will be publicly accessible which could identify you as a participant.

7. **Voluntary Nature of the Study.** If you agree to participate, you are free to withdraw from the study at any time without prejudice. You will be provided a copy of this form for your records.

8. **Points of Contact.** If you have any further questions or comments after the completion of the study, you may contact the research supervisor, LT David Back, (831) 656-3710, dnback@nps.navy.mil.

9. **Statement of Consent.** I have read the above information. I have asked all questions and have had my questions answered. I agree to participate in this study.

--	--------------------------
Participant's Signature	Date
--	--------------------------
Researcher's Signature	Date

MINIMAL RISK CONSENT STATEMENT

NAVAL POSTGRADUATE SCHOOL, MONTEREY, CA 93943
MINIMAL RISK CONSENT STATEMENT

Participant:

VOLUNTARY CONSENT TO BE A RESEARCH PARTICIPANT IN: AGENT-BASED SOLDIER BEHAVIORS IN DYNAMIC VIRTUAL ENVIRONMENTS.

1. I have read, understand and been provided "Information for Participants" that provides the details of the below acknowledgments.
2. I understand that this project involves research. An explanation of the purposes of the research, a description of procedures to be used, identification of experimental procedures, and the extended duration of my participation have been provided to me.
3. I understand that this project does not involve more than minimal risk. I have been informed of any reasonably foreseeable risks or discomforts to me.
4. I have been informed of any benefits to me or to others that may reasonably be expected from the research.
5. I have signed a statement describing the extent to which confidentiality of records identifying me will be maintained.
6. I have been informed of any compensation and/or medical treatments available if injury occurs and is so, what they consist of, or where further information may be obtained.
7. I understand that my participation in this project is voluntary, and refusal to participate will involve no penalty or loss of benefits to which I am otherwise entitled. I also understand that I may discontinue participation at any time without penalty or loss of benefits to which I am otherwise entitled.
8. I understand that the individual to contact should I need answers to pertinent questions about the research is Dr. Michael Capps, Principal Investigator, and about my rights as a research participant or concerning a research related injury is the Modeling Virtual Environments and Simulation Chairman. A full and responsive discussion of the elements of this project and my consent has taken place.

Signature of Principal Investigator	Date	Signature of Volunteer	Date

Signature of Witness	Date

PRIVACY ACT STATMENT

NAVAL POSTGRADUATE SCHOOL, MONTEREY, CA 93943
PRIVACY ACT STATEMENT

1. Authority: Naval Instruction

2. Purpose: DETERMINE FIDELITY OF AGENT-BASED SOLDIER SIMULATION IN DYNAMIC VIRTUAL ENVIRONMENT.

3. Use: Survey data will be used for statistical analysis by the Departments of the Navy and Defense, and other U.S. Government agencies, provided this use is compatible with the purpose for which the information was collected. The Naval Postgraduate School in accordance with the provisions of the Freedom of Information Act may grant use of the information to legitimate non-government agencies or individuals.

4. Disclosure/Confidentiality:

    a. I have been assured that my privacy will be safeguarded. I will be assigned a control or code number, which thereafter will be the only identifying entry on any of the research records. The Principal Investigator will maintain the cross-reference between name and control number. It will be decoded only when beneficial to me or if some circumstances, which are not apparent at this time, would make it clear that decoding would enhance the value of the research data. In all cases, the provisions of the Privacy Act Statement will be honored.

    b. I understand that a record of the information contained in this Consent Statement or derived from the experiment described herein will be retained permanently at the Naval Postgraduate School or by higher authority. I voluntarily agree to its disclosure to agencies or individuals indicated in paragraph 3 and I have been informed that failure to agree to such disclosure may negate the purpose for which the experiment was conducted.

    c. I also understand that disclosure of the requested information, including my Social Security Number, is voluntary.

____

Signature of Volunteer   Name, Grade/Rank (if applicable) DOB   SSN   Date

____

Signature of Witness   Date

Warfare Game
Realism Survey Questionnaire

I. Background Information:

Age: ________
Gender (M/F): ________

For each of the following questions, circle the category that describes you best:

How often do you play First-Person-Shooter style computer games (Doom, Unreal, Quake, etc)?

Never	**Once or Twice**	**Monthly**	**Weekly**	**Daily**

Are you planning on joining the U.S. Army at some point in your life?

No	**Unlikely**	**Possibly**	**Likely**	**Definitely**

II. Game Survey

Which game did you play first (circle one)?

Return to Castle Wolfensteinä **Development Game**

Rate how realistic you though each of the games was in the categories below. 'Realistic' is subjective, but should rate how you compare what you see in the game with what you would expect to see happen in real life in a similar situation. Use the following marks to indicate your answers:

Rating	**Totally unrealistic**	**Mostly unrealistic**	**Moderately realistic**	**Very realistic**	**Totally realistic**
Mark	- -	-	**0**	+	++

Category	**Examples**	**Wolfensteinä**	**Dev**
Responses	Reactions to the player's movements and actions; range of sight and hearing; ability to track the character when out of direct line of sight.		
Appropriateness	Choice of actions in response to the current situation; good choice of tactical decisions based on available information and physical condition		
Self-Preservation Behaviors	Actions take to evade, take cover, and avoid the player when they need to.		
Aggressive Behaviors	Actions taken to track, pursue, and attack the player.		
Other Behaviors	Actions while wandering around, ignoring the player, or unaware of the player.		

Did you notice any behaviors in either game which stood out as particularly realistic or unrealistic? If so, please describe it briefly below:

	Subject Number							
Demographics	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Age	18	18	17	17	17	16	15	15
Gender	F	F	M	F	M	M	M	M
FPS Gaming	Never	Never	Once	Never	Weekly	Weekly	Weekly	Monthly
Army Career	Unlikely	Unlikely	?	Likely	Likely	Possibly	Likely	Possibly
First Played	Wolf	AGP	Wolf	AGP	Wolf	AGP	Wolf	AGP
Wolfenstein								
Response	+	+	+	0	0	-	0	+
Appropriate	++	+	++	+	0	-	+	+
Self-Pres	+	+	++	+	-	0	+	+
Aggressive	- -	+	++	++	+	+	0	0
Other	- -	+	++	+	++	0	0	+
AGP								
Response	+	+	0	-	+	+	+	++
Appropriate	++	+	-	+	0	++	+	0
Self-Pres	+	+	0	++	++	++	++	+
Aggressive	- -	+	0	+	0	++	++	+
Other	- -	0	+	+	+	+	+	+

	Subject Number							
Demographics	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Age	17	15	15	16	16	16	16	18
Gender	M	M	M	F	M	M	M	M
FPS Gaming	Once	Monthly	Daily	Once	Monthly	Daily	Weekly	Weekly
Army Career	Likely	Possibly	Definite	Likely	No	Likely	Possibly	Definite
First Played	Wolf	AGP	Wolf	AGP	Wolf	AGP	Wolf	AGP
Wolfenstein								
Response	+	0	0	++	+	+	+	-
Appropriate	++	0	+	++	0	+	+	0
Self-Pres	0	0	0	+	+	+	0	0
Aggressive	+	+	-	++	+	++	0	0
Other	-	0	+	+	+	0	+	-
AGP								
Response	+	+	++	+	++	++	0	+
Appropriate	+	0	++	++	++	+	+	+
Self-Pres	0	++	++	+	++	+	0	0
Aggressive	++	++	-	++	++	0	0	0
Other	-	+	+	+	++	0	+	0

Table 12. Realism Survey Results

LIST OF REFERENCES

Adobbati, R. and others, (2001), *GameBots: A 3D Virtual World Testbed for Multi-Agent Research*, Proceedings of the 2nd Workshop on Infrastructure for Agents, MAS, and Scalable MAS, ACM Press, New York, 2001, pp 47-52.

Baron-Cohen, S. (1995). *Mindblindness*. MIT Press, 1995.

Calder, R.B., and others, (1993), *ModSAF Behavior Simulation and Control*. Proceedings of the Second Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, July 1993.

Denny, C.R. (2001) *Combat XXI*, [http://tradoc.monroe.army.mil/dcssa/XXI.htm]

Foster, J. (2002), *Close Combat Tactical Trainer XXI*, [http://stricom.army.mil/PRODUCTS/CCTT_XXI/]

Funge, J. (1999), *AI for Computer Games and Animation: A Cognitive Modeling Approach*, AK Peters, Ltd, 1999.

Ilachinski, A. (1997). *Irreducible Semi-Autonomous Adaptive Combat (ISAAC): An Artificial-Life Approach to Land Combat* (U), Center for Naval Analysis Research Memorandum CRM 97-61.10, August 1997, Unclassified.

Ilachinski, A. (1999), *Toward a Science of Experimental Complexity: An Artificial Life Approach to Modeling Warfare*, Presented at 5th Experimental Chaos Conference, Orlando, Florida, 28 June 1999.

Jones, R.M., and others, (1999). *Automated Intelligent Pilots for Combat Flight Simulation*, AI Magazine, 20(1), 27-42

Laird, J.E. (2000). *It Knows What You're Going to Do: Adding Anticipation to a Quakebot*, AAAI 2000 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment, March 2000: AAAI Technical Report SS00-02

Ramsey, B. (2002) *Janus Main Page,* [http://www-leav.army.mil/nsc/famsim/janus/]

Stone, P. and Veloso, M. (2000). *Multiagent systems: A survey from a machine learning perspective.* Autonomous Robots, 8(3), 2000.

Stone, P. (2000). *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press, 2000.

Taylor, J.G. (1980), *Lanchester-Type Models of Warfare*, Technical Report, U.S. Naval Postgraduate School, Monterey CA, 1980.

INITIAL DISTRIBUTION LIST

Defense Technical Information Center
    8725 John J. Kingman Road, Suite 0944
    Ft. Belvoir, VA 22060-6218

Dudley Knox Library
    Naval Postgraduate School
    411 Dyer Road
    Monterey, CA 93943-5101

Professor Michael Zyda
    Code MOVES
    Naval Postgraduate School
    Monterey, CA

Research Assistant Professor Michael Capps
    Code CS/Cm
    Naval Postgraduate School
    Monterey, CA

Research Professor John Hiles
    Code MOVES
    Naval Postgraduate School
    Monterey, CA

LT David Back, USN
    Department Head Course, class 169
    Surface Warfare Officers School
    Newport, RI

Christian Buhl
    Code MOVES
    Naval Postgraduate School
    Monterey, CA